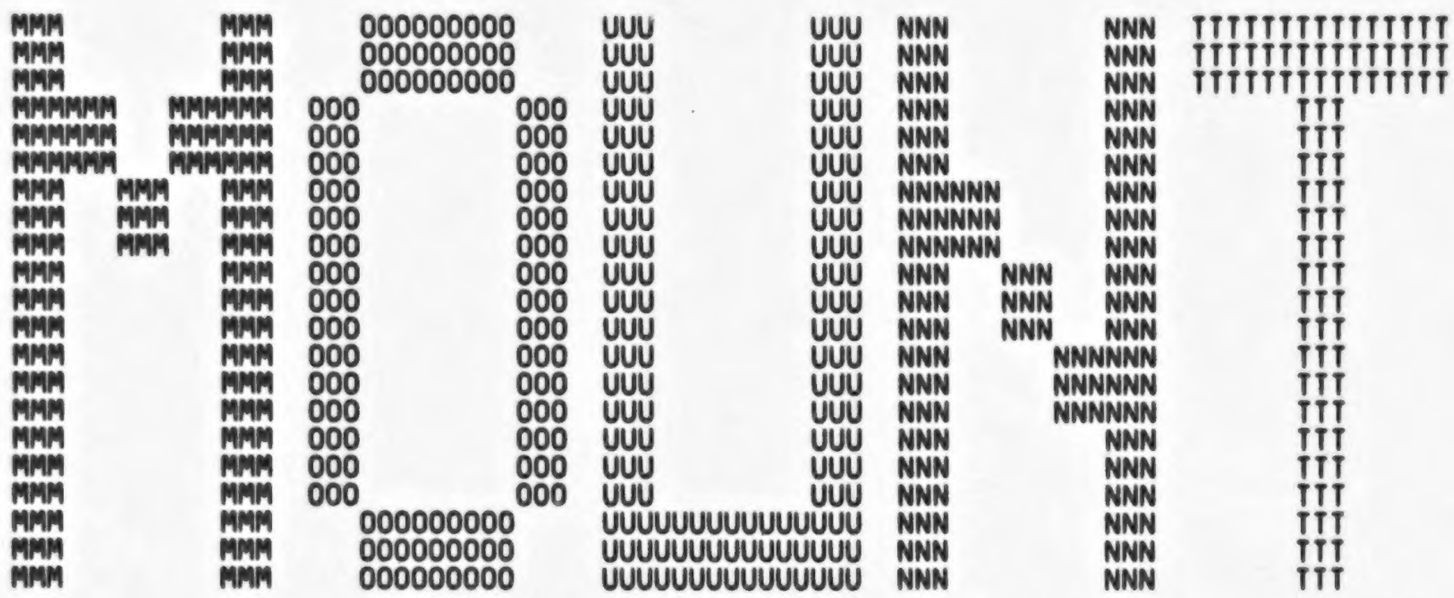
MMM MMM OOOOOOOOO UUU UUU NNN NNN TTTTTTTTTTTTTTT
MMM MMM OOOOOOOOO UUU UUU NNN NNN TTTTTTTTTTTTTTT
MMM MMM OOOOOOOOO UUU UUU NNN NNN TTTTTTTTTTTTTTT
MMMMMM MMMMMM OOO OOO UUU UUU NNN NNN TTT
MMMMMM MMMMMM OOO OOO UUU UUU NNN NNN TTT
MMMMMM MMMMMM OOO OOO UUU UUU NNN NNN TTT
MMM MMM MMM OOO OOO UUU UUU NNNNNN NNN TTT
MMM MMM MMM OOO OOO UUU UUU NNNNNN NNN TTT
MMM MMM MMM OOO OOO UUU UUU NNNNNN NNN TTT
MMM MMM OOO OOO UUU UUU NNN NNN NNN TTT
MMM MMM OOO OOO UUU UUU NNN NNN NNN TTT
MMM MMM OOO OOO UUU UUU NNN NNN NNN TTT
MMM MMM OOO OOO UUU UUU NNN NNNNNN TTT
MMM MMM OOO OOO UUU UUU NNN NNNNNN TTT
MMM MMM OOO OOO UUU UUU NNN NNNNNN TTT
MMM MMM OOO OOO UUU UUU NNN NNN TTT
MMM MMM OOO OOO UUU UUU NNN NNN TTT
MMM MMM OOO OOO UUU UUU NNN NNN TTT
MMM MMM OOOOOOOOO UUUUUUUUUUUUUUU NNN NNN TTT
MMM MMM OOOOOOOOO UUUUUUUUUUUUUUU NNN NNN TTT
MMM MMM OOOOOOOOO UUUUUUUUUUUUUUU NNN NNN TTT

FILEID**MOUTAP

```
MM      MM    OOOOOO    UU      UU  TTTTTTTTTT    AAAAAA    PPPPPPPP
MM      MM    OOOOOO    UU      UU  TTTTTTTTTT    AAAAAA    PPPPPPPP
MMMM  MMMM  OO      OO  UU      UU      TT       AA    AA  PP      PP
MMMM  MMMM  OO      OO  UU      UU      TT       AA    AA  PP      PP
MM  MM  MM  OO      OO  UU      UU      TT       AA    AA  PP      PP
MM  MM  MM  OO      OO  UU      UU      TT       AA    AA  PP      PP
MM      MM  OO      OO  UU      UU      TT       AA    AA  PPPPPPPP
MM      MM  OO      OO  UU      UU      TT       AA    AA  PPPPPPPP
MM      MM  OO      OO  UU      UU      TT       AAAAAAAAAA  PP
MM      MM  OO      OO  UU      UU      TT       AAAAAAAAAA  PP
MM      MM  OO      OO  UU      UU      TT       AA    AA  PP
MM      MM  OO      OO  UU      UU      TT       AA    AA  P?
MM      MM    OOOOOO    UUUUUUUUUU      TT       AA    AA  PP
MM      MM    OOOOOO    UUUUUUUUUU      TT       AA    AA  PP

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

....
....
....
....

```
;      1          0001  0 MODULE MOUTAP (
;      2          0002  0                 LANGUAGE (BLISS32),
;      3          0003  0                 IDENT = 'V04-000'
;      4          0004  0                 ) =
;      5          0005  1 BEGIN
;      6          0006  1
;      7          0007  1 !
;      8          0008  1 !**********************************************************************
;      9          0009  1 !*                                                                    *
;     10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                          *
;     11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.           *
;     12          0012  1 !*  ALL RIGHTS RESERVED.                                            *
;     13          0013  1 !*                                                                    *
;     14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;     15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;     16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
;     17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILAELE TO ANY  *
;     18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
;     19          0019  1 !*  TRANSFERRED.                                                       *
;     20          0020  1 !*                                                                    *
;     21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
;     22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
;     23          0023  1 !*  CORPORATION.                                                       *
;     24          0024  1 !*                                                                    *
;     25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;     26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.            *
;     27          0027  1 !*                                                                    *
;     28          0028  1 !*                                                                    *
;     29          0029  1 !**********************************************************************
;     30          0030  1 !
;     31          0031  1
;     32          0032  1 !++
;     33          0033  1 !
;     34          0034  1 ! FACILITY:  MOUNT Utility
;     35          0035  1 !
;     36          0036  1 ! ABSTRACT:
;     37          0037  1 !
;     38          0038  1 !       These routines handle the mounting of magnetic tape
;     39          0039  1 !
;     40          0040  1 ! ENVIRONMENT:
;     41          0041  1 !
;     42          0042  1 !       VAX/VMS operating system, including privileged system services
;     43          0043  1 !       and internal exec routines.
;     44          0044  1 !
;     45          0045  1 !--
;     46          0046  1 !
;     47          0047  1 !
;     48          0048  1 ! AUTHOR:  D. H. Gillespie,      CREATION DATE:  05-Dec-1977
;     49          0049  1 !
;     50          0050  1 ! MODIFIED BY:
;     51          0051  1 !
;     52          0052  1 !       V03-021 HH0046          Hai Huang                   10-Aug-1984
;     53          0053  1 !               Increment refcount stored in UCB on mount.
;     54          0054  1 !
;     55          0055  1 !       V03-020 HH0041          Hai Huang                   24-Jul-1984
;     56          0056  1 !               Remove REQUIRE 'LIBD$:[VMSLIB.OBJ]MOUNTMSG.B32'.
;     57          0057  1 !
```

```
;    58     0058  1 !       V03-019 HH0035          Hai Huang                10-Jul-1984
;    59     0059  1 !               Fix truncation errors.
;    60     0060  1 !
;    61     0061  1 !       V03-018 MMD0290         Meg Dumont,      10-Apr-1984  14:41
;    62     0062  1 !               Fix to the return from $MTACCESS code were ACCESS could
;    63     0063  1 !               be set to normal processing with out processing all the
;    64     0064  1 !               possible error conditions.
;    65     0065  1 !
;    66     0066  1 !       V03-017 LMP0221         L. Mark Pilant,          28-Mar-1984  10:03
;    67     0067  1 !               Change UCB$L_OWNUIC to ORB$L_OWNER and UCB$W_VPROT to
;    68     0068  1 !               ORB$W_PROT.
;    69     0069  1 !
;    70     0070  1 !       V03-016 MMD0270         Meg Dumont,      23-Mar-1984  9:29
;    71     0071  1 !               Change the processing of the accessibility character fields
;    72     0072  1 !               in the VOL1 and or HDR1 label to call the installation
;    73     0073  1 !               specific accessibility routine. The return from this
;    74     0074  1 !               routine determines the users access to the volume and/or file.
;    75     0075  1 !               This support includes saving the ANSI version number from the
;    76     0076  1 !               VOL1 for future processing of the file header accessibility
;    77     0077  1 !               field.
;    78     0078  1 !
;    79     0079  1 !       V03-015 HH0002          Hai Huang        01-Feb-1984
;    80     0080  1 !               Add job-wide mount support, i.e. always deallocate
;    81     0081  1 !               mount list entry to paged-pool in condition handler.
;    82     0082  1 !
;    83     0083  1 !       V03-014 HH0001          Hai Huang        16-Jan-1984  14:52
;    84     0084  1 !               Fix bug in privilege check code.
;    85     0085  1 !
;    86     0086  1 !       V03-013 MMD0215         Meg Dumont,      3-Jan-1984  16:04
;    87     0087  1 !               Fix bug in protection check code.
;    88     0088  1 !
;    89     0089  1 !       V03-012 MMD0199         Meg Dumont,      25-Aug-1983  10:12
;    90     0090  1 !               Fix bug where if /PROTECTION is specified SYSTEM and
;    91     0091  1 !               OWNER are not given access to the tape. SYSTEM and
;    92     0092  1 !               OWNER should always have access to mounted tapes.
;    93     0093  1 !
;    94     0094  1 !       V03-011 MMD0186         Meg Dumont,      7-Jul-1983  9:59
;    95     0095  1 !               Make the default for AVL/AVR the same from the DCL call
;    96     0096  1 !               and from the system service call.
;    97     0097  1 !
;    98     0098  1 !
;    99     0099  1 !       V03-010 DMW4042         DMWalp                   7-Jun-1983
;   100     0100  1 !               Remove (S)LOG_ENTRY
;   101     0101  1 !
;   102     0102  1 !       V03-009 MMD0179         Meg Dumont,      26-May-1983  15:15
;   103     0103  1 !               Change VOL1 to indicate ANSI level 4 when writing a SYSTEM CODE
;   104     0104  1 !               in the VOL1 label
;   105     0105  1 !
;   106     0106  1 !       V03-008 MMD0136         Meg Dumont,      12-Apr-1983  17:29
;   107     0107  1 !               Added support for writng and interrupting the VOL1
;   108     0108  1 !               OWNER IDENTIFIER field, so that it is no longer
;   109     0109  1 !               treated as a VMS field, strictly. Added support for
;   110     0110  1 !               the underscore as a valid character to tape.
;   111     0111  1 !
;   112     0112  1 !
;   113     0113  1 !       V03-007 MMD0113         Meg Dumont,      29-Mar-1983  0:27
;   114     0114  1 !               Added support for setting AVR, AVL. Added support for new VMS
```

```
;  115     0115  1 !               protection on tape, which includes understanding a VOL2
;  116     0116  1 !               label. Also reformated so that routines common to INIT,
;  117     0117  1 !               MOUNT and the MTAACP could all be shared.
;  118     0118  1 !
;  119     0119  1 !       V03-006 MMD0105         Meg Dumont,     17-Feb-1983  13:25
;  120     0120  1 !               Changed call to CLEAR_VALID to issue a IO$_AVAILABLE
;  121     0121  1 !
;  122     0122  1 !       V03-005 MMD0002         Meg Dumont,     3-Jan-1983  14:50
;  123     0123  1 !               Allow user with read access to tape to mount it writelocked.
;  124     0124  1 !
;  125     0125  1 !       V03-004 MMD0001         Meg Dumont,     13-Aug-1982  13:07
;  126     0126  1 !               Change from call to SET_VALID to QIO IO$_PACKACK
;  127     0127  1 !
;  128     0128  1 !       V03-003 STJ0302         Steven T. Jeffreys,     18-May-1982
;  129     0129  1 !               Add support for /NOUNLOAD qualifier.
;  130     0130  1 !
;  131     0131  1 !       V03-002 STJ0261         Steven T. Jeffreys,     22-Apr-1982
;  132     0132  1 !               Do not mung device allocation access mode.
;  133     0133  1 !               Set the DEADMO bit properly for multi-volume mounts.
;  134     0134  1 !
;  135     0135  1 !       V03-001 STJ0255         Steven T. Jeffreys,     04-Apr-1982
;  136     0136  1 !               Use common I/O routines where possible.
;  137     0137  1 !
;  138     0138  1 !       V02-022 STJ0154         Steven T. Jeffreys,     02-Jan-1981
;  139     0139  1 !               Fix external references to use general addressing mode.
;  140     0140  1 !
;  141     0141  1 !       V02-021 DMW0018         David Michael Walp      17-Dec-1981
;  142     0142  1 !               Increase the size of the translation table to 256
;  143     0143  1 !
;  144     0144  1 !       V02-020 DMW0017         David Michael Walp       3-Dec-1981
;  145     0145  1 !               Return non-ANSI characters as space and fix edit cut
;  146     0146  1 !               and paste error ( wrong index )
;  147     0147  1 !
;  148     0148  1 !       V02-019 DMW0016         David Michael Walp      15-Sep-1981
;  149     0149  1 !               Uppercase and set NOT Unused the MVL entries.
;  150     0150  1 !
;  151     0151  1 !       V02-018 STJ0121         Steven T. Jeffreys      10-Sep-1981
;  152     0152  1 !               Make descriptor references use symbolic offsets.!
;  153     0153  1 !               Checked in a new source.
;  154     0154  1 !
;  155     0155  1 !       V02-017 DMW0015         David Michael Walp      18-Jul-1981
;  156     0156  1 !               Upcase Volume labels, Added 1st Reel Volume Protection
;  157     0157  1 !               and UIC, handles BAD UICs in VOL1
;  158     0158  1 !
;  159     0159  1 !       V02-016 DMW0012         David Michael Walp      30-Jul-1981
;  160     0160  1 !               Store need privilege mask in MVL
;  161     0161  1 !
;  162     0162  1 !       V02-015 DMW0011         David Michael Walp      22-Jul-1981
;  163     0163  1 !               Detect write ring.  Prompted by SPR
;  164     0164  1 !
;  165     0165  1 !       V02-014 DMW0010         David Michael Walp      20-Jul-1981
;  166     0166  1 !               Reset the blocksize when the density is reset.
;  167     0167  1 !
;  168     0168  1 !       V02-013 DMW0009         David Michael Walp      6-Jul-1981
;  169     0169  1 !               Clean up defaulting of density.
;  170     0170  1 !
;  171     0171  1 !       V02-012 DMW0006         David Michael Walp      10-Jun-1981
```

```
;  172     0172  1 !               Major rewrite of MOUNT_TAPE code to allow operator assist
;  173     0173  1 !               to work.  The loop that was in MOUNT_TAPE to ALLOCATE and
;  174     0174  1 !               ASSIGN devices is now in the MOUNT_VOLUME.  READ_VOLLABEL
;  175     0175  1 !               and MOUNT_TAPE may now be called more than once ( if more
;  176     0176  1 !               then a single device is specified to be used ).
;  177     0177  1 !
;  178     0178  1 !       V02-011 DMW0004         David Michael Walp      11-May-1981
;  179     0179  1 !               Stuffed volume access character in MVL and reguire
;  180     0180  1 !               VOLPRO or UIC ownership to MOUNT/FOR an ANSI tape.
;  181     0181  1 !
;  182     0182  1 !       V02-010 DMW0003         David Michael Walp      27-Apr-1981
;  183     0183  1 !               Made "/FOREIGN" and "/NOLABEL" work the same
;  184     0184  1 !
;  185     0185  1 !       V02-009 DMW0002         David Michael Walp      14-Apr-1981
;  186     0186  1 !               Added V3 volume accesiblity code, cleaned up protection holes,
;  187     0187  1 !               added storage of ANSI volume file set id in MVL.
;  188     0188  1 !
;  189     0189  1 !       V02-008 RLR36704                Robert L. Rappaport     2-April-1981
;  190     0190  1 !               Correct the problem of MOUNT returning SS$_VOLINV when
;  191     0191  1 !               the MOUNT command follows a DISMOUNT/NOUNLOAD sequence
;  192     0192  1 !               in a command procedure.
;  193     0193  1 !
;  194     0194  1 !       V02-006 ACG0169         Andrew C. Goldstein,    18-Apr-1980  14:02
;  195     0195  1 !               Bug check on internal errors
;  196     0196  1 !
;  197     0197  1 !       V02-005 ACG0167         Andrew C. Goldstein,    18-Apr-1980  13:38
;  198     0198  1 !               Previous revision history moved to MOUNT.REV
;  199     0199  1 !**
;  200     0200  1
;  201     0201  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;  202     0202  1 REQUIRE 'SRC$:MOUDEF.B32';
;  203     0734  1
;  204     0735  1 FORWARD ROUTINE
;  205     0736  1         ERROR_HANDLER,                          ! handler to clear valid on
;  206     0737  1                                                 !  secondary UCB's
;  207     0738  1         KERNEL_HANDLER  : NOVALUE,              ! kernel mode exception handler
;  208     0739  1         MAKE_TAPE_MOUNT,                        ! kernel moude tape mount
;  209     0740  1         MOUNT_TAPE      : NOVALUE,              ! mount magnetic tape
;  210     0741  1         READ_VOLLABEL,                          ! read and verify VOL1 label
;  211     0742  1         RESET_DENSITY   : NOVALUE,              ! reset the density default
;  212     0743  1         SET_CHARACTER   : NOVALUE;              ! set device characteristics
;  213     0744  1
;  214     0745  1 EXTERNAL ROUTINE
;  215     0746  1         ALLOC_LOGNAME,                          ! allocate logical name
;  216     0747  1         ALLOCATE_MEM,                           ! allocate memory
;  217     0748  1         CHECK_PROT,                             ! check the UIC protection
;  218     0749  1         ENTER_LOGNAME,                          ! enter logical name
;  219     0750  1         GET_CHANNELUCB,                         ! get UCB from channel
;  220     0751  1         GET_RECORD,                             ! get current record drive is reading
;  221     0752  1         LIB$CVT_OTB     : ADDRESSING_MODE (GENERAL),
;  222     0753  1         LOCK_IODB       : ADDRESSING_MODE (GENERAL), ! lock I/O data base
;  223     0754  1         PROCESS_VOL2_LABEL,                     ! process VOL2 label
;  224     0755  1         SEND_ERRLOG,                            ! send message to error logger
;  225     0756  1         START_ACP,                              ! startup ACP
;  226     0757  1         TAPE_OWN_PROT,                          ! determine owner and
;  227     0758  1                                                 !  protection of tape
;  228     0759  1         TRAN_LOGNAME,                           ! translate logical name
```

```
;  229     0760  1          UNLOCK_IODB     : ADDRESSING_MODE (GENERAL); ! unlock I/O database
;  230     0761  1
;  231     0762  1
;  232     0763  1 EXTERNAL
;  233     0764  1          BLOCKSIZE,                                 ! value of /BLOCKSIZE:
;  234     0765  1          CHANNEL,                                   ! channel of tape being mounted
;  235     0766  1          CLEANUP_FLAGS   : BITVECTOR,               ! cleanup flags
;  236     0767  1          CLEANUP_ALLOC   : BITVECTOR,               ! cleanup allocation flags
;  237     0768  1          CTL$GL_VOLUMES  : ADDRESSING_MODE (ABSOLUTE),
;  238     0769  1          DEVICE_CHAR     : BBLOCK,                  ! characteristics of device
;  239     0770  1                                                     !  current being mounted
;  240     0771  1          DEVICE_INDEX    : LONG VOLATILE,           ! index into the device and
;  241     0772  1                                                     !  label lists
;  242     0773  1          MOUNT_OPTIONS   : BITVECTOR,               ! mount option bits
;  243     0774  1          LABEL_COUNT,                               ! number of labels specified
;  244     0775  1          RECORDSZ,                                  ! value of /RECORD:
;  245     0776  1          VOL1            : BBLOCK;                  ! VOL1 label
;  246     0777  1
;  247     0778  1 LITERAL
;  248     0779  1          PROTO_RVT_LEN   = $BYTEOFFSET (RVT$L_UCBLST) + (4*DEVMAX),
;  249     0780  1          PROTO_MVL_LEN   = MVL$K_FIXLEN + (MVL$K_LENGTH*LABMAX);
;  250     0781  1 OWN
;  251     0782  1          ACCESS,                                    ! user's access to magnetic tape
;  252     0783  1          ANSI_LABEL      : BBLOCK [80],             ! buffer to store labels
;  253     0784  1          BLOCKSZ         : WORD,                    ! block size for this volume
;  254     0785  1          FIRST_V_UIC,                               ! owner UIC of 1st tape
;  255     0786  1          FIRST_V_PROT,                              ! 1st tape protection
;  256     0787  1          IO_STATUS       : VECTOR [4,WORD],         ! I/O status block
;  257     0788  1          LABEL_VER,                                 ! decimal ANSI label version
;  258     0789  1          PRIVILEGE_MASK  : REF BBLOCK,              ! user privileges
;  259     0790  1          PROCESS_UIC,                               ! UIC of current process
;  260     0791  1          PROTO_VCB       : BBLOCK[VCB$C_LENGTH]     ! prototype VCB
;  261     0792  1                            INITIAL ( REP VCB$C_LENGTH OF BYTE (0)),
;  262     0793  1          PROTO_RVT       : BBLOCK[PROTO_RVT_LEN]    ! prototype RVT
;  263     0794  1                            INITIAL ( REP PROTO_RVT_LEN OF BYTE (0)),
;  264     0795  1          PROTO_MVL       : BBLOCK[PROTO_MVL_LEN]    ! prototype MVL
;  265     0796  1                            INITIAL ( REP PROTO_MVL_LEN OF BYTE (0)),
;  266     0797  1          VOLUME_PROT,                               ! tape protection
;  267     0798  1          VOLUME_UIC,                                ! owner UIC of tape
;  268     0799  1          WRITE_RING      : BITVECTOR [ 1 ];         ! are any write rings missing
;  269     0800  1
;  270     0801  1 BIND
;  271     0802  1          STARID = UPLIT ('DECFILE11A'),
;  272     0803  1
;  273     0804  1          ! UPLIT was used instead of CH$TRANSTABLE here, the code
;  274     0805  1          !  produced is the same (ie the constant string generated).
;  275     0806  1          !  UPLIT was used because CH$TRANSTABLE generates a warning error
;  276     0807  1          !  because more then a single character at a time is specified
;  277     0808  1          !  in the %ASCII. ( BLISS KLUDGE )
;  278     0809  1          !
;  279     0810  1          ! The table will upcase a..z and return space for any non ANSI
;  280     0811  1          !  'a' characters.
;  281     0812  1          !
;  282     0813  1          TRANSLATION_TABLE = UPLIT BYTE (
;  283     0814  1                  %ASCII '                                ',
;  284     0815  1                  %ASCII ' !"  %&''()*+,-./0123456789:;<=>?',
;  285     0816  1                  %ASCII ' ABCDEFGHIJKLMNOPQRSTUVWXYZ    _',
```

```
;  286          0817  1          %ASCII '  ABCDEFGHIJKLMNOPQRSTUVWXYZ      ',
;  287          0818  1          %ASCII '                                  ',
;  288          0819  1          %ASCII '                                  ',
;  289          0820  1          %ASCII '                                  ',
;  290          0821  1          %ASCII '                                  ');
```

```
;  292     0822  1 GLOBAL ROUTINE READ_VOLLABEL (VOLUME_LABEL)  =
;  293     0823  1
;  294     0824  1 !++
;  295     0825  1 !
;  296     0826  1 ! FUNCTIONAL DESCRIPTION:
;  297     0827  1 !
;  298     0828  1 !       This routine reads the first block on the magnetic tape and
;  299     0829  1 !       checks that it is an ANSI tape
;  300     0830  1 !
;  301     0831  1 ! CALLING SEQUENCE:
;  302     0832  1 !       READ_VOLLABEL (ARG1)
;  303     0833  1 !
;  304     0834  1 ! INPUT PARAMETERS:
;  305     0835  1 !       ARG1 - address of volume label string descriptor
;  306     0836  1 !
;  307     0837  1 ! IMPLICIT INPUTS:
;  308     0838  1 !       CHANNEL - channel number assigned to device being mounted
;  309     0839  1 !
;  310     0840  1 ! OUTPUT PARAMETERS:
;  311     0841  1 !       NONE
;  312     0842  1 !
;  313     0843  1 ! IMPLICIT OUTPUTS:
;  314     0844  1 !       VOL1            - VOL1 magnetic tape label
;  315     0845  1 !       VOLUME_UIC      - owner of tape
;  316     0846  1 !       VOLUME_PROT     - tape protection
;  317     0847  1 !
;  318     0848  1 ! ROUTINE VALUE:
;  319     0849  1 !       SS$_NORMAL      - if valid ANSI volume label
;  320     0850  1 !       SS$_NOTLABELMT  - not labeled ANSI magnetic tape
;  321     0851  1 !       SS$_INCVOLLABEL - incorrect volume label
;  322     0852  1 !       SS$_DEVOFFLINE  - device not on system
;  323     0853  1 !       SS$_MEDOFL      - medium off_line
;  324     0854  1 !
;  325     0855  1 ! SIDE EFFECTS:
;  326     0856  1 !       NONE
;  327     0857  1 !
;  328     0858  1 ! USER ERRORS:
;  329     0859  1 !       NONE
;  330     0860  1 !
;  331     0861  1 !--
;  332     0862  1
;  333     0863  2 BEGIN
;  334     0864  2
;  335     0865  2 EXTERNAL
;  336     0866  2         CTL$GL_PHD      : REF BBLOCK ADDRESSING_MODE(ABSOLUTE);
;  337     0867  2
;  338     0868  2 MAP
;  339     0869  2         VOLUME_LABEL    : REF BBLOCK;   ! volume label ( from command line )
;  340     0870  2                                         !   string desc
;  341     0871  2
;  342     0872  2 LOCAL
;  343     0873  2     CURRENT_RECORD,                     ! Current record the tape drive is reading
;  344     0874  2     UCB : REF BBLOCK,                   ! Address of ucb
;  345     0875  2     UPCASE_INPUT    : VECTOR [ VL1$S_VOLLBL, BYTE ],
;  346     0876  2     UPCASE_TAPE     : VECTOR [ VL1$S_VOLLBL, BYTE ],
;  347     0877  2     TAPE_OWNER_STS,
;  348     0878  2     STATUS,
```

```
;  349          0879  2      VMS_TAPE;                       ! Set if VMS created tape
;  350          0880  2
;  351          0881  2  BIND
;  352          0882  2      SECONDS         = UPLIT (-10000000,-1);     ! one second in 100 nsec units
;  353          0883  2
;  354          0884  2  ! Enable handler to clear valid on all but current device
;  355          0885  2  !
;  356          0886  2  ENABLE ERROR_HANDLER;
;  357          0887  2  !
;  358          0888  2  ! The following is here for historical reasons only
;  359          0889  2  !*****************************************************************************
;  360          0890  2  ! Here we have inserted two extra QIO's (IO$_REWIND) which apparently are not
;  361          0891  2  !  needed but which in fact are here to take care of an anomaly that
;  362          0892  2  !  sometimes occurs when the MOUNT command appears in a command file
;  363          0893  2  !  immediately following a DISMOUNT/NOUNLOAD command.
;  364          0894  2  !
;  365          0895  2  !  Under certain circumstances the MOUNT fails with a SS$_VOLINV status.
;  366          0896  2  !  The problem is due to a complicated interaction involving QIO dispatching
;  367          0897  2  !  logic, the MAGTAPE ACP, and the MOUNT command.  What occurs is the
;  368          0898  2  !  following.
;  369          0899  2  !
;  370          0900  2  !  DISMOUNT, before finishing issues a $QIOW with an I/O function code of
;  371          0901  2  !  IO$_ACPCONTROL!IO$M_DMOUNT.  This request is forwarded to the ACP and
;  372          0902  2  !  DISMOUNT then has its image rundown.
;  373          0903  2  !
;  374          0904  2  !  The ACP then issues a $QIOW with a function code of IO$_REWIND!IO$M_NOWAIT,
;  375          0905  2  !  while in parallel, MOUNT is starting up and it proceeds to set the
;  376          0906  2  !  UCB$M_VALID bit in UCB$W_STS (which in this case was still on due to the
;  377          0907  2  !  volume previously having been mounted) and then MOUNT issues its own
;  378          0908  2  !  $QIOW with an IO$_REWIND funciton code.
;  379          0909  2  !
;  380          0910  2  !  In some instances, the ACP's REWIND QIO does not get as far as REQCOM
;  381          0911  2  !  until after MOUNT's REWIND has been queued.  If this occurs, INIT's
;  382          0912  2  !  queued REWIND is started up before the ACP actually regains control and
;  383          0913  2  !  the driver has no trouble since it finds the UCB$M_VALID bit still on.
;  384          0914  2  !  Unfortunately, as soon as the ACP regains control, following the
;  385          0915  2  !  driver's WFIKPCH, the ACP clears the UCB$M_VALID bit.  The next QIO
;  386          0916  2  !  issued by MOUNT will fail due to the absence of the UCB$M_VALID
;  387          0917  2  !  bit.
;  388          0918  2  !
;  389          0919  2  !  The solution (pronounced KLUDGE) herein implemented, simply inserts an extra
;  390          0920  2  !  couple of $QIOW's with IO$_REWIND function code, preceeded by explicit
;  391          0921  2  !  settings of the UCB$M_VALID bit, before the real logic of MOUNT begins.
;  392          0922  2  !  These $QIOW's allow the above potential interaction to occur, and after they
;  393          0923  2  !  have finished, we again set the UCB$M_VALID bit on in the normal way.
;  394          0924  2  !*****************************************************************************
;  395          0925  2  !
;  396          0926  2  ! The above is no longer true; that is we have elimentated the race condition
;  397          0927  2  ! mentioned above by not doing issuing the rewind at dismount time
;  398          0928  2  ! but infact marking the drive available. The following IO's mark
;  399          0929  2  ! the volume valid then issue the rewind, which is neccesary because
;  400          0930  2  ! of the preMSCP drivers will not rewind on this function. The MSCP drivers
;  401          0931  2  ! will and the second IO here becomes an NOP.
;  402          0932  2  !
;  403          0933  2  !
;  404          0934  2
;  405        P 0935  2  STATUS = DO_IO(
```

```
;  406     P 0936  2          CHAN = .CHANNEL,
;  407     P 0937  2          FUNC = IO$_PACKACK,
;  408       0938  2          IOSB = IO_STATUS[0]);
;  409       0939  2
;  410       0940  2
;  411     P 0941  2  STATUS = DO_IO(
;  412     P 0942  2          CHAN = .CHANNEL,
;  413     P 0943  2          FUNC = IO$_REWIND,
;  414       0944  2          IOSB = IO_STATUS[0]);
;  415       0945  2
;  416       0946  2
;  417       0947  2  ! some things which need to be set up only the first time thru
;  418       0948  2
;  419       0949  2  IF .DEVICE_INDEX EQL 0
;  420       0950  2  THEN
;  421       0951  2      BEGIN
;  422       0952  2
;  423       0953  2      ! Assume that the user is correct on the command line about write ring
;  424       0954  2      !   status
;  425       0955  2
;  426       0956  2      WRITE_RING [ 0 ] = .MOUNT_OPTIONS [ OPT_WRITE ];
;  427       0957  2
;  428       0958  2      ! get the UIC of the current process
;  429       0959  2
;  430       0960  2      $GETJPI ( ITMLST = UPLIT( WORD(4), WORD(JPI$_UIC), LONG(PROCESS_UIC,0,0)));
;  431       0961  2
;  432       0962  2      ! determine user's privilege from process privilege mask
;  433       0963  2
;  434       0964  2      PRIVILEGE_MASK = CTL$GL_PHD[PHD$Q_PRIVMSK];
;  435       0965  2
;  436       0966  2      END;
;  437       0967  2
;  438       0968  2  ! Set up the device characteristics.  This can be done even if the drive is
;  439       0969  2  ! offline.
;  440       0970  2
;  441       0971  2  SET_CHARACTER ();
;  442       0972  2
;  443       0973  2  ! Set up the default volume UIC and volume protection.  Default UIC is the
;  444       0974  2  ! current process.  This is done in case this is a non-ANSI tape or more device
;  445       0975  2  ! then labels have been specified. Default protection is no world or
;  446       0976  2  ! group access to the tape.
;  447       0977  2
;  448       0978  2  VOLUME_UIC  = .PROCESS_UIC;
;  449       0979  2  VOLUME_PROT = 0;
;  450       0980  2
;  451       0981  2  ! If there are more devices then labels specified then exit here because we
;  452       0982  2  ! can not check a label if we do not know it.  This does not matter if it is
;  453       0983  2  ! the first time thru because the label must be specified or /OVER=ID used
;  454       0984  2  ! ( in which case we will return the label )
;  455       0985  2
;  456       0986  2  IF     (.DEVICE_INDEX NEQ 0)
;  457       0987  2    AND (.DEVICE_INDEX GEQ .LABEL_COUNT)
;  458       0988  2    AND NOT ( .MOUNT_OPTIONS [ OPT_FOREIGN ] OR .MOUNT_OPTIONS [ OPT_NOLABEL ] )
;  459       0989  2  THEN RETURN SS$_NORMAL;
;  460       0990  2
;  461       0991  2  ! Position tape to BOT and check status
;  462       0992  2  ! wait 10 seconds before deciding that the device is offline
```

```
;  463          0993  2
;  464          0994  2 INCRU J FROM 0 TO 9 DO
;  465          0995  3     BEGIN
;  466        P 0996  3         STATUS = DO_IO(
;  467        P 0997  3                  CHAN = .CHANNEL,
;  468        P 0998  3                  FUNC = IO$_PACKACK,
;  469          0999  3                  IOSB = IO_STATUS[0]);
;  470        P 1000  3     STATUS = DO_IO (    CHAN = .CHANNEL,
;  471        P 1001  3                         FUNC = IO$_REWIND,
;  472          1002  3                         IOSB = IO_STATUS);
;  473          1003  3     IF .STATUS THEN STATUS = .IO_STATUS[0];
;  474          1004  3     IF .STATUS NEQ SS$_MEDOFL AND .STATUS NEQ SS$_VOLINV THEN EXITLOOP;
;  475          1005  4     IF $SETIMR (REQIDT = 999, DAYTIM = SECONDS, EFN = TIMER_EFN)
;  476          1006  3     THEN
;  477          1007  4         BEGIN
;  478          1008  4         $WAITFR (EFN = TIMER_EFN);
;  479          1009  4         $CANTIM (REQIDT = 999);
;  480          1010  4         $SETEF  (EFN = TIMER_EFN);
;  481          1011  3         END;
;  482          1012  2     END;
;  483          1013  2
;  484          1014  2 ! All errors other than device not in system or medium off line reported
;  485          1015  2 !  to user
;  486          1016  2
;  487          1017  2 IF NOT .STATUS THEN ERR_EXIT (.STATUS);
;  488          1018  2
;  489          1019  2 ! Test to see if the write ring is really there, only if we think it should
;  490          1020  2 ! be there.
;  491          1021  2
;  492          1022  2 IF .WRITE_RING [ 0 ]
;  493          1023  2 THEN
;  494          1024  3     BEGIN
;  495          1025  3
;  496          1026  3     ! allow us to get at the information nicely
;  497          1027  3
;  498          1028  3     BIND DEVICE_DEPENDENT = IO_STATUS [ 2 ] : BBLOCK;
;  499          1029  3
;  500        P 1030  3     STATUS = DO_IO (CHAN = .CHANNEL,
;  501        P 1031  3                     IOSB = IO_STATUS,
;  502          1032  3                     FUNC = IO$_SENSEMODE);
;  503          1033  3     IF .STATUS THEN STATUS = .IO_STATUS[0];
;  504          1034  3     IF NOT .STATUS THEN ERR_EXIT (.STATUS);
;  505          1035  3
;  506          1036  3     ! NOTE: assignment done only if we think a write ring should be there
;  507          1037  3
;  508          1038  3     WRITE_RING [ 0 ] = NOT (.DEVICE_DEPENDENT [ MT$V_HWL ]);
;  509          1039  3
;  510          1040  3     END;
;  511          1041  2
;  512          1042  2 ! Do not read the tape if /FORIEGN and /OVER=(ACC,EXP), VOLPRO and OPER
;  513          1043  2 ! This allows the operator to initialize blank tapes.
;  514          1044  2 !  ( run away tape condition with brand new tapes )
;  515          1045  2 ! Please note that this really is a hack to allow operators to get around the
;  516          1046  2 ! fact that some hardware can not deal with blank tapes. This should
;  517          1047  2 ! not be the defacto for initializing tapes.
;  518          1048  2
;  519          1049  2 IF   .PRIVILEGE_MASK [ PRV$V_VOLPRO ]
```

```
;  520          1050  2    AND .PRIVILEGE_MASK [ PRV$V_OPER ]
;  521          1051  2    AND ( .MOUNT_OPTIONS [ OPT_FOREIGN ] OR .MOUNT_OPTIONS [ OPT_NOLABEL ] )
;  522          1052  2    AND .MOUNT_OPTIONS [ OPT_OVR_ACC ]
;  523          1053  2    AND .MOUNT_OPTIONS [ OPT_OVR_EXP ]
;  524          1054  2  THEN RETURN SS$_NORMAL;
;  525          1055  2
;  526          1056  2  ! Read first block on tape and check status
;  527          1057  2
;  528        P 1058  2  STATUS = DO_IO (CHAN = .CHANNEL,
;  529        P 1059  2                  FUNC = IO$_READLBLK,
;  530        P 1060  2                  IOSB = IO_STATUS,
;  531        P 1061  2                  P1 = VOL1,
;  532        P 1062  2                  P2 = 80);
;  533          1063  2  IF .STATUS THEN STATUS = .IO_STATUS[0];
;  534          1064  2
;  535          1065  2  ! If first record is TM then not ANSI tape
;  536          1066  2  ! If label is more than 80 characters ignore error
;  537          1067  2
;  538          1068  2  IF (NOT .STATUS) AND (.STATUS NEQ SS$_DATAOVERUN)
;  539          1069  2  THEN
;  540          1070  3      BEGIN
;  541          1071  3      RESET_DENSITY ();
;  542          1072  3      RETURN SS$_NOTLABELMT;
;  543          1073  3      END;
;  544          1074  2
;  545          1075  2  ! Now check that first block is VOL1 ANSI label
;  546          1076  2
;  547          1077  2  IF .VOL1[VL1$L_VL1LID] NEQ 'VOL1'
;  548          1078  2  THEN
;  549          1079  3      BEGIN
;  550          1080  3      RESET_DENSITY ();
;  551          1081  3      RETURN SS$_NOTLABELMT;
;  552          1082  3      END;
;  553          1083  2
;  554          1084  2  ! determine owner and VMS protection of tape
;  555          1085  2
;  556          1086  2  TAPE_OWNER_STS = TAPE_OWN_PROT (VOLUME_UIC, VOLUME_PROT, .PROCESS_UIC, VOL1);
;  557          1087  2
;  558          1088  2  ! Get the ANSI version from the label and subtract the character 0 to
;  559          1089  2  ! make it a decimal value rather than ASCII. Use the channel to get the
;  560          1090  2  ! physical UCB.
;  561          1091  2
;  562          1092  2  LABEL_VER = .VOL1[VL1$B_LBLSTDVER] - '0';
;  563          1093  2  UCB = KERNEL_CALL(GET_CHANNELUCB, .CHANNEL);
;  564          1094  2
;  565          1095  2  ! Call the accessibility system service to check the accessibility char
;  566          1096  2  ! on the VOL1 label.
;  567          1097  2  ! First keep the record that the UCB is reading. The accessibility
;  568          1098  2  ! routine can not move the tape from under us! Thus we will compare
;  569          1099  2  ! this to the field after the call and if the tape was moved we punt
;  570          1100  2  ! the operation. Grant the user access to the volume according to
;  571          1101  2  ! the error code returned from the system service.
;  572          1102  2
;  573          1103  2  CURRENT_RECORD = KERNEL_CALL(GET_RECORD, .UCB);
;  574        P 1104  2  ACCESS = $MTACCESS(LBLNAM = VOL1,
;  575        P 1105  2                     UIC = .PROCESS_UIC,
;  576        P 1106  2                     STD_VERSION = .LABEL_VER,
```

```
;   577        P 1107  2                   ACCESS_CHAR = 0,
;   578        P 1108  2                   ACCESS_SPEC = MTA$K_NOCHAR,
;   579          1109  2                   TYPE = MTA$K_INVOL1);
;   580          1110  2 STATUS = KERNEL_CALL(GET_RECORD, .UCB);
;   581          1111  2 IF .CURRENT_RECORD NEQ .STATUS
;   582          1112  2    THEN ERR_EXIT(SS$_TAPEPOSLOST);
;   583          1113  2
;   584          1114  2 ! Now check the ACCESS returned from the service. For SS$_FILACCERR
;   585          1115  2 ! check to make sure /OVERRIDE=ACCESS was specified and the user
;   586          1116  2 ! has privilege then set to check VMS protection.
;   587          1117  2 ! For SS$_NOFILACC, SS$_NOVOLACC return the code
;   588          1118  2 ! to the user. In this case the user has no access to the tape volume.
;   589          1119  2 ! For a 0 give the user all access. For SS$_NORMAL check the VMS
;   590          1120  2 ! protection.
;   591          1121  2
;   592          1122  2 IF .ACCESS EQL SS$_NOVOLACC
;   593          1123  2    OR .ACCESS EQL SS$_NOFILACC
;   594          1124  2         THEN ERR_EXIT(.ACCESS);
;   595          1125  2
;   596          1126  2 IF .ACCESS EQL SS$_FILACCERR
;   597          1127  2    THEN
;   598          1128  3         BEGIN
;   599          1129  3            IF NOT .MOUNT_OPTIONS[OPT_OVR_ACC]
;   600          1130  3                 THEN ERR_EXIT(.ACCESS);
;   601          1131  3            IF NOT .PRIVILEGE_MASK[PRV$V_VOLPRO]
;   602          1132  3                 THEN ERR_EXIT(.ACCESS);
;   603          1133  3            ACCESS = SS$_NORMAL;
;   604          1134  3         END;
;   605          1135  2
;   606          1136  2
;   607          1137  2 ! If ACCESS is 0 then the user has full access to the tape regardless
;   608          1138  2 ! of the VMS protection specified
;   609          1139  2
;   610          1140  2 IF NOT .ACCESS THEN VOLUME_PROT = 0;
;   611          1141  2
;   612          1142  2 ! If tape was created by VMS then the system code should match that
;   613          1143  2 ! of VMS. If the system code does not then do not process the VOL2
;   614          1144  2 ! label
;   615          1145  2
;   616          1146  2 IF CH$EQL(10,STARID,10,VOL1[VL1ST_SYSCODE],0)
;   617          1147  2    THEN VMS_TAPE = 1
;   618          1148  2    ELSE VMS_TAPE = 0;
;   619          1149  2
;   620          1150  2 ! first record on tape is VOL1.  The next record may be a VOL2
;   621          1151  2 ! label if it is then process it otherwise process the HDR1 label.
;   622          1152  2 ! NOTE:  User volume labels may intervene.
;   623          1153  2
;   624          1154  2 WHILE 1 DO
;   625          1155  3     BEGIN
;   626        P 1156  3     STATUS = DO_IO(
;   627        P 1157  3         CHAN = .CHANNEL,
;   628        P 1158  3         FUNC = IO$_READLBLK,
;   629        P 1159  3         IOSB = IO_STATUS[0],
;   630        P 1160  3         P1 = ANSI_LABEL,
;   631          1161  3         P2 = 80);
;   632          1162  3     IF .STATUS THEN STATUS = .IO_STATUS[0];
;   633          1163  3
```

```
;  634          1164  3      ! ANSI tape, but can't read HDR1
;  635          1165
;  636          1166  3      IF NOT .STATUS AND (.STATUS NEQ SS$_DATAOVERUN) THEN RETURN SS$_NOTLABELMT;
;  637          1167
;  638          1168  3      ! If the SYSCODE was VMS's and there is a VOL2 Label then process it.
;  639          1169  3      ! After processing the VOL2 label we must check the ACCESS field so that
;  640          1170  3      ! if the accessibility routine gave the user full access to the volume
;  641          1171  3      ! then the VMS protection must be set up so that the user has full
;  642          1172  3      ! access to the volume.
;  643          1173  3
;  644          1174  3      IF .VMS_TAPE AND .ANSI_LABEL[VL2$L_VL2LID] EQL 'VOL2'
;  645          1175  3          THEN
;  646          1176  4          BEGIN
;  647          1177  4              PROCESS_VOL2_LABEL(VOLUME_UIC, VOLUME_PROT, .PROCESS_UIC,
;  648          1178  4                                 ANSI_LABEL);
;  649          1179  4              IF NOT .ACCESS THEN VOLUME_PROT = 0;
;  650          1180  4          END;
;  651          1181  3      IF .ANSI_LABEL[HD1$L_HD1LID] EQL 'HDR1' THEN EXITLOOP;
;  652          1182  3      END;
;  653          1183  2
;  654          1184  2  ! Must have VOLPRO privilege or UIC ownership to mount a ANSI tape /foreign
;  655          1185  2
;  656          1186  2  IF ( .MOUNT_OPTIONS [ OPT_FOREIGN ] OR .MOUNT_OPTIONS [ OPT_NOLABEL ] )
;  657          1187  2  THEN
;  658          1188  3      BEGIN
;  659          1189  4      IF (.PRIVILEGE_MASK [ PRV$V_VOLPRO ]) OR (.PROCESS_UIC EQL .VOLUME_UIC)
;  660          1190  3      THEN
;  661          1191  4          BEGIN
;  662          1192  4          RESET_DENSITY ();
;  663          1193  4          RETURN SS$_NORMAL;
;  664          1194  4          END
;  665          1195  3      ELSE RETURN SS$_NOPRIV;
;  666          1196  2      END;
;  667          1197  2
;  668          1198  2
;  669          1199  2  ! If the owner identifier field of the VOL1 label can not allow the user to
;  670          1200  2  ! access the tape with out forceing the user to specify /OVERRIDE=OWNER_ID
;  671          1201  2  ! and the accessibility routine specified to check VMS protection than
;  672          1202  2  ! punt the MOUNT.
;  673          1203  2
;  674          1204  2      IF NOT .TAPE_OWNER_STS AND NOT .MOUNT_OPTIONS[OPT_OVR_VOLO] AND .ACCESS
;  675          1205  2          THEN ERR_EXIT (SS$_VOLOERR);
;  676          1206  2
;  677          1207  2  ! Now check if the labels match.  First, test the length of the input string
;  678          1208  2
;  679          1209  2  IF .VOLUME_LABEL [DSC$W_LENGTH] GTRU VL1$S_VOLLBL THEN ERR_EXIT (SS$_MTLBLLONG);
;  680          1210  2
;  681          1211  2  ! Next translate the labels into uppercase and put in ' ' for any non-ANSI
;  682          1212  2  !  "'a' characters" found.  Pad with space, in case the label from command is
;  683          1213  2  !  less than six characters long.
;  684          1214  2  !
;  685          1215  2  CH$TRANSLATE (TRANSLATION_TABLE, .VOLUME_LABEL[DSC$W_LENGTH], .VOLUME_LABEL[DSC$A_POINTER], ' ',
;  686          1216  2                                  VL1$S_VOLLBL,     UPCASE_INPUT);
;  687          1217  2  CH$TRANSLATE (TRANSLATION_TABLE, VL1$S_VOLLBL,     VOL1[VL1$T_VOLLBL], ' ',
;  688          1218  2                                  VL1$S_VOLLBL,     UPCASE_TAPE);
;  689          1219  2
;  690          1220  2  IF CH$NEQ (VL1$S_VOLLBL, UPCASE_INPUT, VL1$S_VOLLBL, UPCASE_TAPE)
```

```
;  691          1221  2 THEN RETURN SS$_INCVOLLABEL;
;  692          1222  2
;  693          1223  2 RETURN SS$_NORMAL;
;  694          1224  1 END;                                    ! end of routine READ_VOLLABEL

                                                                .TITLE  MOUTAP
                                                                .IDENT  \V04-000\

                                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

                00 00 41 31 31 45 4C 49 46 43 45 44  00000 P.AAA:  .ASCII  \DECFILE11A\<0><0>
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20  0000C P.AAB:  .ASCII  \                \
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20  0001B
                                                 20 20  0002A
2E 2D 2C 2B 2A 29 28 27 26 25 20 20 22 21 20 20  0002C          .ASCII  \ !"  %&'()*+,-./0123456789:;<=>?\
3D 3C 3B 3A 39 38 37 36 35 34 33 32 31 30 2F 20  0003B
                                                 3F 3E  0004A
4E 4D 4C 4B 4A 49 48 47 46 45 44 43 42 41 20 20  0004C          .ASCII  \ ABCDEFGHIJKLMNOPQRSTUVWXYZ    _\
20 20 20 5A 59 58 57 56 55 54 53 52 51 50 4F 20  0005B
                                                 5F 20  0006A
4E 4D 4C 4B 4A 49 48 47 46 45 44 43 42 41 20 20  0006C          .ASCII  \ ABCDEFGHIJKLMNOPQRSTUVWXYZ     \
20 20 20 5A 59 58 57 56 55 54 53 52 51 50 4F 20  0007B
                                                 20 20  0008A
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20  0008C          .ASCII  \                \
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20  0009B
                                                 20 20  000AA
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20  000AC          .ASCII  \                \
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20  000BB
                                                 20 20  000CA
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20  000CC          .ASCII  \                \
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20  000DB
                                                 20 20  000EA
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20  000EC          .ASCII  \                \
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20  000FB
                                                 20 20  0010A
                                FFFFFFFF  FF676980  0010C P.AAC:  .LONG   -10000000, -1
                                     0004  00114 P.AAD:  .WORD   4
                                     0304  00116         .WORD   772
                              00000000'    00118         .ADDRESS PROCESS_UIC
                                00000000  00000000  0011C         .LONG   0, 0

                                                                .PSECT  $OWN$,NOEXE,2

                                                 00000 ACCESS:  .BLKB   4
                                                 00004 ANSI_LABEL:
                                                       .BLKB   80
                                                 00054 BLOCKSZ: .BLKB   2
                                                 00056          .BLKB   2
                                                 00058 FIRST_V_UIC:
                                                       .BLKB   4
                                                 0005C FIRST_V_PROT:
                                                       .BLKB   4
                                                 00060 IO_STATUS:
                                                       .BLKB   8
                                                 00068 LABEL_VER:
                                                       .BLKB   4
```

```
                00060 PRIVILEGE_MASK:
                                .BLKB   4
                00070 PROCESS_UIC:
                                .BLKB   4
            00# 00074 PROTO_VCB:
                                .BYTE   0[236]                                  ;
            00# 00160 PROTO_RVT:
                                .BYTE   0[132]                                  ;
            00# 001E4 PROTO_MVL:
                                .BYTE   0[164]                                  ;
                00288 VOLUME_PROT:
                                .BLKB   4
                0028C VOLUME_UIC:
                                .BLKB   4
                00290 WRITE_RING:
                                .BLKB   1

                                STARID=                 P.AAA
                                TRANSLATION_TABLE=      P.AAB
                                SECONDS=                P.AAC
                                DEVICE_DEPENDENT=       IO_STATUS+4
                                        .EXTRN  ALLOC_LOGNAME, ALLOCATE_MEM
                                        .EXTRN  CHECK_PROT, ENTER_LOGNAME
                                        .EXTRN  GET_CHANNELUCB, GET_RECORD
                                        .EXTRN  LIB$CVT_OTB, LOCK_IODB
                                        .EXTRN  PROCESS_VOL2_LABEL
                                        .EXTRN  SEND_ERRLOG, START_ACP
                                        .EXTRN  TAPE_OWN_PROT, TRAN_LOGNAME
                                        .EXTRN  UNLOCK_IODB, BLOCKSIZE
                                        .EXTRN  CHANNEL, CLEANUP_FLAGS
                                        .EXTRN  CLEANUP_ALLOC, CTL$GL_VOLUMES
                                        .EXTRN  DEVICE_CHAR, DEVICE_INDEX
                                        .EXTRN  MOUNT_OPTIONS, LABEL_COUNT
                                        .EXTRN  RECORDSZ, VOL1, CTL$GL_PHD
                                        .EXTRN  COMMON_IO, SYS$GETJPI
                                        .EXTRN  SYS$SETIMR, SYS$WAITFR
                                        .EXTRN  SYS$CANTIM, SYS$SETEF
                                        .EXTRN  SYS$CMKRNL, SYS$MTACCESS

                                        .PSECT  $CODE$,NOWRT,2

                            0FFC 00000          .ENTRY  READ_VOLLABEL, Save R2,R3,R4,R5,R6,R7,R8,-      ; 0822
                                                        R9,R10,R11
5B      0000G   CF   9E 00002                   MOVAB   VOL1, R11                                       ;
5A      0000G   CF   9E 00007                   MOVAB   CHANNEL, R10                                    ;
59      0000G   CF   9E 0000C                   MOVAB   MOUNT_OPTIONS, R9                               ;
58  00000000G   00   9E 00011                   MOVAB   COMMON_IO, R8                                   ;
57  00000000G   00   9E 00018                   MOVAB   LIB$STOP, R7                                    ;
56      0000'   CF   9E 0001F                   MOVAB   IO_STATUS, R6                                   ;
5E              10   C2 00024                   SUBL2   #16, SP                                         ;
6D      0353    CF   DE 00027                   MOVAL   38$, (FP)                                       ; 0863
                7E   7C 0002C                   CLRQ    -(SP)                                           ; 0938
                7E   7C 0002E                   CLRQ    -(SP)                                           ;
                7E   7C 00030                   CLRQ    -(SP)                                           ;
                7E   7C 00032                   CLRQ    -(SP)                                           ;
                56   DD 00034                   PUSHL   R6                                              ;
                08   DD 00036                   PUSHL   #8                                              ;
```

```
                                        6A  DD 00038          PUSHL   CHANNEL
                                        1A  DD 0003A          PUSHL   #26
                          68            0C  FB 0003C          CALLS   #12, COMMON_IO
                          54            50  D0 0003F          MOVL    R0, STATUS
                                        7E  7C 00042          CLRQ    -(SP)                                     ; 0944
                                        7E  7C 00044          CLRQ    -(SP)
                                        7E  7C 00046          CLRQ    -(SP)
                                        7E  7C 00048          CLRQ    -(SP)
                                        56  DD 0004A          PUSHL   R6
                                        24  DD 0004C          PUSHL   #36
                                        6A  DD 0004E          PUSHL   CHANNEL
                                        1A  DD 00050          PUSHL   #26
                          68            0C  FB 00052          CALLS   #12, COMMON_IO
                          54            50  D0 00055          MOVL    R0, STATUS
                                 0000G  CF  D5 00058          TSTL    DEVICE_INDEX                              ; 0949
                                        28  12 0005C          BNEQ    1$
       50       01   A9           01    01  EF 0005E          EXTZV   #1, #1, MOUNT_OPTIONS+1, R0               ; 0956
  0230 C6            01           00    50  F0 00064          INSV    R0, #0, #1, WRITE_RING
                                        7E  7C 0006B          CLRQ    -(SP)                                     ; 0960
                                        7E  D4 0006D          CLRL    -(SP)
                                 0000'  CF  9F 0006F          PUSHAB  P.AAD
                                        7E  7C 00073          CLRQ    -(SP)
                                        7E  D4 00075          CLRL    -(SP)
                   00000000G      00    07  FB 00077          CALLS   #7, SYS$GETJPI
                          0C      A6 00000000G 9F D0 0007E    MOVL    @#CTL$GL_PHD, PRIVILEGE_MASK              ; 0964
                        0000V     CF    00  FB 00086  1$:     CALLS   #0, SET_CHARACTER                         ; 0971
                        022C      C6    10  A6 D0 0008B       MOVL    PROCESS_UIC, VOLUME_UIC                   ; 0978
                                 0228   C6  D4 00091          CLRL    VOLUME_PROT                               ; 0979
                                 0000G  CF  D5 00095          TSTL    DEVICE_INDEX                              ; 0986
                                        16  13 00099          BEQL    2$
                        0000G     CF     0000G  CF D1 0009B   CMPL    DEVICE_INDEX, LABEL_COUNT                 ; 0987
                                        0D  19 000A2          BLSS    2$
                   08       01    A9    03  E0 000A4          BBS     #3, MOUNT_OPTIONS+1, 2$                   ; 0988
                   03       01    A9    04  E0 000A9          BBS     #4, MOUNT_OPTIONS+1, 2$
                                      02C9  31 000AE          BRW     37$
                                        52  D4 000B1  2$:     CLRL    J                                         ; 0994
                                        7E  7C 000B3  3$:     CLRQ    -(SP)                                     ; 0999
                                        7E  7C 000B5          CLRQ    -(SP)
                                        7E  7C 000B7          CLRQ    -(SP)
                                        7E  7C 000B9          CLRQ    -(SP)
                                        56  DD 000BB          PUSHL   R6
                                        08  DD 000BD          PUSHL   #8
                                        6A  DD 000BF          PUSHL   CHANNEL
                                        1A  DD 000C1          PUSHL   #26
                          68            0C  FB 000C3          CALLS   #12, COMMON_IO
                          54            50  D0 000C6          MOVL    R0, STATUS
                                        7E  7C 000C9          CLRQ    -(SP)                                     ; 1002
                                        7E  7C 000CB          CLRQ    -(SP)
                                        7E  7C 000CD          CLRQ    -(SP)
                                        7E  7C 000CF          CLRQ    -(SP)
                                        56  DD 000D1          PUSHL   R6
                                        24  DD 000D3          PUSHL   #36
                                        6A  DD 000D5          PUSHL   CHANNEL
                                        1A  DD 000D7          PUSHL   #26
                          68            0C  FB 000D9          CALLS   #12, COMMON_IO
                          54            50  D0 000DC          MOVL    R0, STATUS
                          03            54  E9 000DF          BLBC    STATUS, 4$                                ; 1003
```

```
                                   54            66  3C 000E2        MOVZWL  IO_STATUS, STATUS                                ;
                        000001A4   8F            54  D1 000E5 4$:    CMPL    STATUS, #420                                     ; 1004
                                   09            13  000EC          BEQL    5$                                               ;
                        00000254   8F            54  D1 000EE        CMPL    STATUS, #596                                     ;
                                   41            12  000F5          BNEQ    7$                                               ;
                                   7E   03E7     8F  3C 000F7 5$:    MOVZWL  #999, -(SP)                                      ; 1005
                                                 7E  D4 000FC        CLRL    -(SP)                                            ;
                                        0000'    CF  9F 000FE        PUSHAB  SECONDS                                          ;
                                                 19  DD 00102        PUSHL   #25                                              ;
                        00000000G  00            04  FB 00104        CALLS   #4, SYS$SETIMR                                   ;
                                   20            50  E9 0010B        BLBC    R0, 6$                                           ;
                                                 19  DD 0010E        PUSHL   #25                                              ; 1008
                        00000000G  00            01  FB 00110        CALLS   #1, SYS$WAITFR                                   ;
                                                 7E  D4 00117        CLRL    -(SP)                                            ; 1009
                                   7E   03E7     8F  3C 00119        MOVZWL  #999, -(SP)                                      ;
                        00000000G  00            02  FB 0011E        CALLS   #2, SYS$CANTIM                                   ;
                                                 19  DD 00125        PUSHL   #25                                              ; 1010
                        00000000G  00            01  FB 00127        CALLS   #1, SYS$SETEF                                    ;
                                                 52  D6 0012E 6$:    INCL    J                                                ; 0994
                                   09            52  D1 00130        CMPL    J, #9                                            ;
                                                 03  1A 00133        BGTRU   7$                                               ;
                                               FF7B  31 00135        BRW     3$                                               ;
                                   05            54  E8 00138 7$:    BLBS    STATUS, 8$                                       ; 1017
                                                 54  DD 0013B        PUSHL   STATUS                                           ;
                                   67            01  FB 0013D        CALLS   #1, LIB$STOP                                     ;
                                   34   0230     C6  E9 00140 8$:    BLBC    WRITE_RING, 11$                                  ; 1022
                                                 7E  7C 00145        CLRQ    -(SP)                                            ; 1032
                                                 7E  7C 00147        CLRQ    -(SP)                                            ;
                                                 7E  7C 00149        CLRQ    -(SP)                                            ;
                                                 7E  7C 0014B        CLRQ    -(SP)                                            ;
                                                 56  DD 0014D        PUSHL   R6                                               ;
                                                 27  DD 0014F        PUSHL   #39                                              ;
                                                 6A  DD 00151        PUSHL   CHANNEL                                          ;
                                                 1A  DD 00153        PUSHL   #26                                              ;
                                   68            0C  FB 00155        CALLS   #12, COMMON_IO                                   ;
                                   54            50  D0 00158        MOVL    R0, STATUS                                       ;
                                   06            54  E9 0015B        BLBC    STATUS, 9$                                       ; 1033
                                   54            66  3C 0015E        MOVZWL  IO_STATUS, STATUS                                ;
                                   05            54  E8 00161        BLBS    STATUS, 10$                                      ; 1034
                                                 54  DD 00164 9$:    PUSHL   STATUS                                           ;
                                   67            01  FB 00166        CALLS   #1, LIB$STOP                                     ;
          50        06     A6      01            03  EF 00169 10$:   EXTZV   #3, #1, DEVICE_DEPENDENT+2, R0                   ; 1038
                                   50            50  D2 0016F        MCOML   R0, R0                                           ;
 0230     C6               01      00            50  F0 00172        INSV    R0, #0, #1, WRITE_RING                           ;
                           1C      0C   B6       15  E1 00179 11$:   BBC     #21, @PRIVILEGE_MASK, 13$                        ; 1049
                           17      0C   B6       12  E1 0017E        BBC     #18, @PRIVILEGE_MASK, 13$                        ; 1050
                           05      01   A9       03  E0 00183        BBS     #3, MOUNT_OPTIONS+1, 12$                         ; 1051
                           0D      01   A9       04  E1 00188        BBC     #4, MOUNT_OPTIONS+1, 13$                         ;
                           08      04   A9       06  E1 0018D 12$:   BBC     #6, MOUNT_OPTIONS+4, 13$                         ; 1052
                           03      02   A9       04  E1 00192        BBC     #4, MOUNT_OPTIONS+2, 13$                         ; 1053
                                               01E0  31 00197        BRW     37$                                              ;
                                                 7E  7C 0019A 13$:   CLRQ    -(SP)                                            ; 1062
                                                 7E  7C 0019C        CLRQ    -(SP)                                            ;
                                   7E     50     8F  9A 0019E        MOVZBL  #80, -(SP)                                       ;
                                                 5B  DD 001A2        PUSHL   R11                                              ;
                                                 7E  7C 001A4        CLRQ    -(SP)                                            ;
                                                 56  DD 001A6        PUSHL   R6                                               ;
```

```
                                 21   DD 001A8          PUSHL   #33                                                    ;
                                 6A   DD 001AA          PUSHL   CHANNEL                                                ;
                                 1A   DD 001AC          PUSHL   #26                                                    ;
                           68    0C   FB 001AE          CALLS   #12, COMMON_IO                                         ;
                           54    50   D0 001B1          MOVL    R0, STATUS                                             ;
                           06    54   E9 001B4          BLBC    STATUS, 14$                                            ; 1063
                           54    66   3C 001B7          MOVZWL  IO_STATUS, STATUS                                      ;
                           09    54   E8 001BA          BLBS    STATUS, 15$                                            ; 1068
00000838                   8F    54   D1 001BD 14$:      CMPL    STATUS, #2104                                          ;
                                 09   12 001C4          BNEQ    16$                                                    ;
314C4F56                   8F    6B   D1 001C6 15$:      CMPL    VOL1, #827084630                                       ; 1077
                                 08   13 001CD          BEQL    17$                                                    ;
                     0000V       CF   00   FB 001CF 16$:      CALLS   #0, RESET_DENSITY                              ; 1080
                                      0102 31 001D4          BRW     28$                                            ; 1081
                                 5B   DD 001D7 17$:      PUSHL   R11                                                    ; 1086
                              10    A6   DD 001D9          PUSHL   PROCESS_UIC                                            ;
                            0228    C6   9F 001DC          PUSHAB  VOLUME_PROT                                            ;
                            022C    C6   9F 001E0          PUSHAB  VOLUME_UIC                                             ;
                     0000G       CF   04   FB 001E4          CALLS   #4, TAPE_OWN_PROT                              ;
                           55    50   D0 001E9          MOVL    R0, TAPE_OWNER_STS                                     ;
                        08 A6    4F   AB 9A 001EC          MOVZBL  VOL1+79, LABEL_VER                             ; 1092
                        08 A6         30   C2 001F1          SUBL2   #48, LABEL_VER                                 ;
                                 6A   DD 001F5          PUSHL   CHANNEL                                                ; 1093
                                 01   DD 001F7          PUSHL   #1                                                     ;
                                 5E   DD 001F9          PUSHL   SP                                                     ;
                            0000G     CF   9F 001FB          PUSHAB  GET_CHANNELUCB                                         ;
00000000G                  9F    04   FB 001FF          CALLS   #4, @#SYS$CMKRNL                                       ;
                           52    50   D0 00206          MOVL    R0, UCB                                                ;
                                 52   DD 00209          PUSHL   UCB                                                    ; 1103
                                 01   DD 0020B          PUSHL   #1                                                     ;
                                 5E   DD 0020D          PUSHL   SP                                                     ;
                            0000G     CF   9F 0020F          PUSHAB  GET_RECORD                                             ;
00000000G                  9F    04   FB 00213          CALLS   #4, @#SYS$CMKRNL                                       ;
                           53    50   D0 0021A          MOVL    R0, CURRENT_RECORD                                     ;
                                 7E   7C 0021D          CLRQ    -(SP)                                                  ; 1109
                                 7E   D4 0021F          CLRL    -(SP)                                                  ;
                              08 A6    DD 00221          PUSHL   LABEL_VER                                              ;
                              10 A6    DD 00224          PUSHL   PROCESS_UIC                                            ;
                                 5B   DD 00227          PUSHL   R11                                                    ;
00000000G                  00    06   FB 00229          CALLS   #6, SYS$MTACCESS                                       ;
                        A0 A6    50   D0 00230          MOVL    R0, ACCESS                                             ;
                                 52   DD 00234          PUSHL   UCB                                                    ; 1110
                                 01   DD 00236          PUSHL   #1                                                     ;
                                 5E   DD 00238          PUSHL   SP                                                     ;
                            0000G     CF   9F 0023A          PUSHAB  GET_RECORD                                             ;
00000000G                  9F    04   FB 0023E          CALLS   #4, @#SYS$CMKRNL                                       ;
                           54    50   D0 00245          MOVL    R0, STATUS                                             ;
                           54    53   D1 00248          CMPL    CURRENT_RECORD, STATUS                                 ; 1111
                                 08   13 0024B          BEQL    18$                                                    ;
                           7E    0224 8F   3C 0024D          MOVZWL  #548, -(SP)                                    ; 1112
                           67    01   FB 00252          CALLS   #1, LIB$STOP                                           ;
                           50    A0   A6   D0 00255 18$:      MOVL    ACCESS, R0                                     ; 1122
000022A4                   8F    50   D1 00259          CMPL    R0, #8868                                              ;
                                 09   13 00260          BEQL    19$                                                    ;
000022AC                   8F    50   D1 00262          CMPL    R0, #8876                                              ; 1123
                                 05   12 00269          BNEQ    20$                                                    ;
                                 50   DD 0026B 19$:      PUSHL   R0                                                     ; 1124
```

```
                       67             01  FB  0026D         CALLS   #1, LIB$STOP                              ;
          0000009C     8F      A0     A6  D1  00270  20$:   CMPL    ACCESS, #156                              ; 1126
                                      1A  12  00278         BNEQ    23$                                       ;
      06        04     A9             06  E0  0027A         BBS     #6, MOUNT_OPTIONS+4, 21$                  ; 1129
                               A0     A6  DD  0027F         PUSHL   ACCESS                                    ; 1130
                       67             01  FB  00282         CALLS   #1, LIB$STOP                              ;
      06        0C     B6             15  E0  00285  21$:   BBS     #21, @PRIVILEGE_MASK, 22$                 ; 1131
                               A0     A6  DD  0028A         PUSHL   ACCESS                                    ; 1132
                       67             01  FB  0028D         CALLS   #1, LIB$STOP                              ;
                A0     A6             01  D0  00290  22$:   MOVL    #1, ACCESS                                ; 1133
                       04      A0     A6  E8  00294  23$:   BLBS    ACCESS, 24$                               ; 1140
                             0228     C6  D4  00298         CLRL    VOLUME_PROT                               ;
 18   AB     0000'     CF             0A  29  0029C  24$:   CMPC3   #10, STARID, VOL1+24                      ; 1146
                                      05  12  002A3         BNEQ    25$                                       ;
                       52             01  D0  002A5         MOVL    #1, VMS_TAPE                              ; 1147
                                      02  11  002A8         BRB     26$                                       ;
                                      52  D4  002AA  25$:   CLRL    VMS_TAPE                                  ; 1148
                                      7E  7C  002AC  26$:   CLRQ    -(SP)                                     ; 1161
                                      7E  7C  002AE         CLRQ    -(SP)                                     ;
                       7E      50     8F  9A  002B0         MOVZBL  #80, -(SP)                                ;
                               A4     A6  9F  002B4         PUSHAB  ANSI_LABEL                                ;
                                      7E  7C  002B7         CLRQ    -(SP)                                     ;
                                      56  DD  002B9         PUSHL   R6                                        ;
                                      21  DD  002BB         PUSHL   #33                                       ;
                                      6A  DD  002BD         PUSHL   CHANNEL                                   ;
                                      1A  DD  002BF         PUSHL   #26                                       ;
                       68             0C  FB  002C1         CALLS   #12, COMMON_IO                            ;
                       54             50  D0  002C4         MOVL    R0, STATUS                                ;
                       06             54  E9  002C7         BLBC    STATUS, 27$                               ; 1162
                       54             66  3C  002CA         MOVZWL  IO_STATUS, STATUS                         ;
                       0F             54  E8  002CD         BLBS    STATUS, 29$                               ; 1166
          00000838     8F             54  D1  002D0  27$:   CMPL    STATUS, #2104                             ;
                                      06  13  002D7         BEQL    29$                                       ;
                       50    01DC     8F  3C  002D9  28$:   MOVZWL  #476, R0                                  ;
                                          04  002DE         RET                                               ;
                       25             52  E9  002DF  29$:   BLBC    VMS_TAPE, 30$                             ; 1174
          324C4F56     8F      A4     A6  D1  002E2         CMPL    ANSI_LABEL, #843861846                    ;
                                      1B  12  002EA         BNEQ    30$                                       ;
                               A4     A6  9F  002EC         PUSHAB  ANSI_LABEL                                ; 1177
                               10     A6  DD  002EF         PUSHL   PROCESS_UIC                               ;
                             0228     C6  9F  002F2         PUSHAB  VOLUME_PROT                               ;
                             022C     C6  9F  002F6         PUSHAB  VOLUME_UIC                                ;
              0000G    CF             04  FB  002FA         CALLS   #4, PROCESS_VOL2_LABEL                    ;
                       04      A0     A6  E8  002FF         BLBS    ACCESS, 30$                               ; 1179
                             0228     C6  D4  00303         CLRL    VOLUME_PROT                               ;
          31524448     8F      A4     A6  D1  00307  30$:   CMPL    ANSI_LABEL, #827475016                    ; 1181
                                      9B  12  0030F         BNEQ    26$                                       ;
      05        01     A9             03  E0  00311         BBS     #3, MOUNT_OPTIONS+1, 31$                  ; 1186
      18        01     A9             04  E1  00316         BBC     #4, MOUNT_OPTIONS+1, 34$                  ;
      08        0C     B6             15  E0  0031B  31$:   BBS     #21, @PRIVILEGE_MASK, 32$                 ; 1189
              022C     C6      10     A6  D1  00320         CMPL    PROCESS_UIC, VOLUME_UIC                   ;
                                      07  12  00326         BNEQ    33$                                       ;
              0000V    CF             00  FB  00328  32$:   CALLS   #0, RESET_DENSITY                         ; 1192
                                      4B  11  0032D         BRB     37$                                       ; 1195
                       50             24  D0  0032F  33$:   MOVL    #36, R0                                   ;
                                          04  00332         RET                                               ;
                       11             55  E8  00333  34$:   BLBS    TAPE_OWNER_STS, 35$                       ; 1204
```

```
                     0C      07  A9              04  E0 00336          BBS     #4, MOUNT_OPTIONS+7, 35$                                ;
                             08      A0      A6  E9 0033B          BLBC    ACCESS, 35$                                             ;
                             7E  226C        8F  3C 0033F          MOVZWL  #8812, -(SP)                                            ; 1205
                             67              01  FB 00344          CALLS   #1, LIB$STOP                                            ;
                             52      04      AC  D0 00347 35$:     MOVL    VOLUME_LABEL, R2                                        ; 1209
                             06              62  B1 0034B          CMPW    (R2), #6                                                ;
                                             08  1B 0034E          BLEQU   36$                                                     ;
                             7E  0304        8F  3C 00350          MOVZWL  #772, -(SP)                                             ;
                             67              01  FB 00355          CALLS   #1, LIB$STOP                                            ;
    0000'  CF        20  04  B2              62  2E 00358 36$:     MOVTC   (R2), @4(R2), #32, TRANSLATION_TABLE, #6, -             ; 1215
                         08  AF              06      00360          UPCASE_INPUT                                                    ;
    0000'  CF        20  04  AB              06  2E 00363          MOVTC   #6, VOL1+4, #32, TRANSLATION_TABLE, #6, -                ; 1217
                             6E              06      00368          UPCASE_TAPE                                                     ;
                     6E      08  AE          06  29 0036D          CMPC3   #6, UPCASE_INPUT, UPCASE_TAPE                           ; 1220
                                             06  13 00372          BEQL    37$                                                     ;
                             50      010C    8F  3C 00374          MOVZWL  #268, R0                                                ; 1221
                                             04  00379          RET                                                             ;
                             50              01  D0 0037A 37$:     MOVL    #1, R0                                                  ; 1223
                                             04  0037D          RET                                                             ; 1224
                                         0000    0037E 38$:     .WORD   Save nothing                                            ; 0863
                                         7E  D4  00380          CLRL    -(SP)                                                   ;
                                         5E  DD  00382          PUSHL   SP                                                      ;
                             7E      04  AC  7D  00384          MOVQ    4(AP), -(SP)                                            ;
                     0000V   CF          03  FB  00388          CALLS   #3, ERROR_HANDLER                                       ;
                                             04  0038D          RET                                                             ;

; Routine Size:  910 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  696     1225  1 ROUTINE SET_CHARACTER : NOVALUE =
;  697     1226  1
;  698     1227  1 !++
;  699     1228  1 !
;  700     1229  1 ! FUNCTIONAL DESCRIPTION:
;  701     1230  1 !
;  702     1231  1 !       This routine sets the tape drive characteristics.
;  703     1232  1 !
;  704     1233  1 ! CALLING SEQUENCE:
;  705     1234  1 !       SET_CHARACTER ();
;  706     1235  1 !
;  707     1236  1 ! INPUT PARAMETERS:
;  708     1237  1 !       NONE
;  709     1238  1 !
;  710     1239  1 ! IMPLICIT INPUTS:
;  711     1240  1 !       DEVICE_CHAR     - The current device characteristics
;  712     1241  1 !       MOUNT_OPTIONS   - The mount option specified by the user
;  713     1242  1 !       BLOCKSIZE       - value of "/BLOCKSIZE"
;  714     1243  1 !       RECORDSIZE      - value of "/RECORDSIZE"
;  715     1244  1 !       CHANNEL         - the I/O channel of the tape drive
;  716     1245  1 !
;  717     1246  1 ! OUTPUT PARAMETERS:
;  718     1247  1 !       NONE
;  719     1248  1 !
;  720     1249  1 ! IMPLICIT OUTPUTS:
;  721     1250  1 !       IO_STATUS       - set to the return status of the QIO
;  722     1251  1 !
;  723     1252  1 ! ROUTINE VALUE:
;  724     1253  1 !       NONE
;  725     1254  1 !
;  726     1255  1 ! SIDE EFFECTS:
;  727     1256  1 !       NONE
;  728     1257  1 !
;  729     1258  1 ! USER ERRORS:
;  730     1259  1 !       NONE
;  731     1260  1 !
;  732     1261  1 !--
;  733     1262  1
;  734     1263  2 BEGIN
;  735     1264  2
;  736     1265  2 LITERAL
;  737     1266  2         ODD_PARITY      = 0;
;  738     1267  2
;  739     1268  2 LOCAL
;  740     1269  2         CHARACTERISTIC  : VECTOR [4,WORD],       ! characteristics to set
;  741     1270  2         STATUS;
;  742     1271  2
;  743     1272  2 BIND
;  744     1273  2         !  Set up offsets into the characteristics buffer
;  745     1274  2
;  746     1275  2         FORMAT          = CHARACTERISTIC[2] : BBLOCK,
;  747     1276  2         PARITY          = CHARACTERISTIC[2] : BBLOCK,
;  748     1277  2         BUFFER_SIZE     = CHARACTERISTIC[1] : WORD,
;  749     1278  2         DENSITY         = CHARACTERISTIC[2] : BBLOCK;
;  750     1279  2
;  751     1280  2 !
;  752     1281  2 ! Initialize characteristics
```

```
;  753     1282  2 !
;  754     1283  2
;  755     1284  2 CHARACTERISTIC[0] = .(DEVICE_CHAR + 4);
;  756     1285  2 CHARACTERISTIC[1] = .(DEVICE_CHAR + 6);
;  757     1286  2 CHARACTERISTIC[2] = .(DEVICE_CHAR + 8);
;  758     1287  2 CHARACTERISTIC[3] = .(DEVICE_CHAR + 10);
;  759     1288  2
;  760     1289  2 ! Now set density
;  761     1290  2
;  762     1291  2 IF .MOUNT_OPTIONS[OPT_DENSITY] THEN
;  763     1292  3     BEGIN
;  764     1293  3     IF .MOUNT_OPTIONS[OPT_DENS_800]
;  765     1294  3       THEN DENSITY[MT$V_DENSITY] = MT$K_NRZI_800
;  766     1295  3       ELSE
;  767     1296  3         IF .MOUNT_OPTIONS[OPT_DENS_1600]
;  768     1297  3           THEN DENSITY[MT$V_DENSITY] = MT$K_PE_1600
;  769     1298  3           ELSE DENSITY[MT$V_DENSITY] = MT$K_GCR_6250;
;  770     1299  3     END
;  771     1300  3     ELSE
;  772     1301  3
;  773     1302  3     ! use the default 1600 BPI
;  774     1303  3
;  775     1304  3     DENSITY[MT$V_DENSITY] = MT$K_PE_1600;
;  776     1305  3
;  777     1306  3
;  778     1307  2
;  779     1308  2 ! Parity set to odd, we only support 9-tracks and 9-tracks are always odd
;  780     1309  2
;  781     1310  2 PARITY [ MT$V_PARITY ] = ODD_PARITY;
;  782     1311  2
;  783     1312  2 ! Reset Tape format to FILES-11 ( only supported format )
;  784     1313  2
;  785     1314  2 FORMAT [ MT$V_FORMAT ] = MT$K_NORMAL11;
;  786     1315  2
;  787     1316  2 !
;  788     1317  2 ! record and block sizes only for mount ( not init )
;  789     1318  2 !
;  790     1319  2
;  791     1320  2 ! Determine block size to set
;  792     1321  2
;  793     1322  2 IF ( .MOUNT_OPTIONS[OPT_FOREIGN] OR .MOUNT_OPTIONS[OPT_NOLABEL] )
;  794     1323  2     THEN BLOCKSZ = 512
;  795     1324  2     ELSE BLOCKSZ = 2048;
;  796     1325  2
;  797     1326  2 ! Check that blocksize for mounted labeled tape is not less than 18
;  798     1327  2
;  799     1328  2 IF .MOUNT_OPTIONS[OPT_BLOCKSIZE] THEN
;  800     1329  3     BEGIN
;  801     1330  3     IF      NOT .MOUNT_OPTIONS[OPT_FOREIGN]
;  802     1331  3        AND NOT .MOUNT_OPTIONS[OPT_NOLABEL]
;  803     1332  3        AND .BLOCKSIZE LSS 18
;  804     1333  3       THEN ERR_EXIT (MOUN$_ILLANSIBS);
;  805     1334  3     BLOCKSZ = .BLOCKSIZE;
;  806     1335  2     END;
;  807     1336  2
;  808     1337  2 BUFFER_SIZE = .BLOCKSZ;
;  809     1338  2
```

```
;  810          1339  2 ! Check legal record size
;  811          1340  2
;  812          1341  2 IF .RECORDSZ GTRU .BLOCKSZ THEN ERR_EXIT (MOUN$_RECGTRBL);
;  813          1342  2
;  814          1343  2 ! write the characteristics to the tape drive
;  815          1344  2
;  816        P 1345  2 STATUS = DO_IO (CHAN = .CHANNEL,
;  817        P 1346  2                 IOSB = IO_STATUS,
;  818        P 1347  2                 FUNC = IO$_SETMODE,
;  819          1348  2                 P1 = CHARACTERISTIC);
;  820          1349  2 IF .STATUS THEN STATUS = .IO_STATUS[0];
;  821          1350  2 IF (NOT .STATUS) AND (.DEVICE_INDEX LSS .LABEL_COUNT) THEN ERR_EXIT (.STATUS);
;  822          1351  2
;  823          1352  1 END;                              ! end of Routine SET_CHARACTER


                                        001C 00000 SET_CHARACTER:
                                                           .WORD   Save R2,R3,R4                                   ; 1225
                                54 00000000G   00   9E 00002       MOVAB   LIB$STOP, R4                            ;
                                53      0000'  CF   9E 00009       MOVAB   BLOCKSZ, R3                             ;
                                52      0000G  CF   9E 0000E       MOVAB   MOUNT_OPTIONS, R2                       ;
                                7E      0000G  CF   7D 00013       MOVQ    DEVICE_CHAR+4, CHARACTERISTIC           ; 1284
                                19             62   E9 00018       BLBC    MOUNT_OPTIONS, 2$                       ; 1291
                          08            62             01   E1 0001B       BBC     #1, MOUNT_OPTIONS, 1$           ; 1293
     05   AE                05            00             03   F0 0001F       INSV    #3, #0, #5, DENSITY+1           ; 1294
                                           13             11   00025       BRB     3$                              ;
                          08    05      A2             03   E0 00027 1$:   BBS     #3, MOUNT_OPTIONS+5, 2$         ; 1296
     05   AE                05            00             05   F0 0002C       INSV    #5, #0, #5, DENSITY+1           ; 1298
                                           06             11   00032       BRB     3$                              ; 1291
     05   AE                05            00             04   F0 00034 2$:   INSV    #4, #0, #5, DENSITY+1           ; 1304
                          04    AE                      08   8A 0003A 3$:   BICB2   #8, PARITY                      ; 1310
     04   AE                04            04             0C   F0 0003E       INSV    #12, #4, #4, FORMAT             ; 1314
                          05    01      A2             03   E0 00044       BBS     #3, MOUNT_OPTIONS+1, 4$         ; 1322
                          07    01      A2             04   E1 00049       BBC     #4, MOUNT_OPTIONS+1, 5$         ;
                                63      0200   8F      B0   0004E 4$:      MOVW    #512, BLOCKSZ                   ; 1323
                                           05   11 00053                   BRB     6$                              ;
                                63      0800   8F      B0   00055 5$:      MOVW    #2048, BLOCKSZ                  ; 1324
                                1F        02   A2      E9   0005A 6$:      BLBC    MOUNT_OPTIONS+2, 8$             ; 1328
                          15    01      A2             03   E0 0005E       BBS     #3, MOUNT_OPTIONS+1, 7$         ; 1330
                          10    01      A2             04   E0 00063       BBS     #4, MOUNT_OPTIONS+1, 7$         ; 1331
                                12      0000G  CF      D1   00068       CMPL    BLOCKSIZE, #18                  ; 1332
                                           09   18 0006D                   BGEQ    7$                              ;
                                   0072B0DC    8F      DD   0006F       PUSHL   #7504092                        ; 1333
                                64             01      FB   00075       CALLS   #1, LIB$STOP                    ;
                                63      0000G  CF      B0   00078 7$:      MOVW    BLOCKSIZE, BLOCKSZ              ; 1334
                          02            AE             63   B0 0007D 8$:   MOVW    BLOCKSZ, BUFFER_SIZE            ; 1337
0000G CF                  63            10             00   ED 00081       CMPZV   #0, #16, BLOCKSZ, RECORDSZ      ; 1341
                                           09   1E 00088                   BGEQU   9$                              ;
                                   0072813C    8F      DD   0008A       PUSHL   #7504188                        ;
                                64             01      FB   00090       CALLS   #1, LIB$STOP                    ;
                                           7E   7C 00093 9$:      CLRQ    -(SP)                                   ; 1348
                                           7E   7C 00095          CLRQ    -(SP)                                   ;
                                           7E   D4 00097          CLRL    -(SP)                                   ;
                                     14    AE   9F 00099          PUSHAB  CHARACTERISTIC                          ;
```

```
                                 7E  7C 0009C           CLRQ    -(SP)                                                          ;
                           0C    A3  9F 0009E           PUSHAB  IO_STATUS                                                      ;
                                 23  DD 000A1           PUSHL   #35                                                            ;
                           0000G CF  DD 000A3           PUSHL   CHANNEL                                                        ;
                                 1A  DD 000A7           PUSHL   #26                                                            ;
                00000000G  00      0C  FB 000A9           CALLS   #12, COMMON_IO                                             ;
                           07      50  E9 000B0           BLBC    STATUS, 10$                                                ; 1349
                           50    0C  A3  3C 000B3           MOVZWL  IO_STATUS, STATUS                                          ;
                           0E      50  E8 000B7           BLBS    STATUS, 11$                                                ; 1350
                     0000G CF      0000G CF  D1 000BA 10$:    CMPL    DEVICE_INDEX, LABEL_COUNT                              ;
                                       05  18 000C1           BGEQ    11$                                                    ;
                                       50  DD 000C3           PUSHL   STATUS                                                 ;
                                 64        01  FB 000C5           CALLS   #1, LIB$STOP                                       ;
                                           04  000C8 11$:      RET                                                            ; 1352

; Routine Size:  201 bytes,    Routine Base:  $CODE$ + 038E
```

```
;  825     1353  1 ROUTINE RESET_DENSITY : NOVALUE =
;  826     1354  1
;  827     1355  1 !++
;  828     1356  1 !
;  829     1357  1 ! FUNCTIONAL DESCRIPTION:
;  830     1358  1 !
;  831     1359  1 !       This routine resets the density of the tape drive.  It is called
;  832     1360  1 !       if this a foreign mount.
;  833     1361  1 !
;  834     1362  1 ! CALLING SEQUENCE:
;  835     1363  1 !       RESET_DENSITY ();
;  836     1364  1 !
;  837     1365  1 ! INPUT PARAMETERS:
;  838     1366  1 !       NONE
;  839     1367  1 !
;  840     1368  1 ! IMPLICIT INPUTS:
;  841     1369  1 !       CHANNEL         - the I/O channel of the tape drive
;  842     1370  1 !
;  843     1371  1 ! OUTPUT PARAMETERS:
;  844     1372  1 !       NONE
;  845     1373  1 !
;  846     1374  1 ! IMPLICIT OUTPUTS:
;  847     1375  1 !       IO_STATUS       - set to the return status of the QIO
;  848     1376  1 !
;  849     1377  1 ! ROUTINE VALUE:
;  850     1378  1 !       NONE
;  851     1379  1 !
;  852     1380  1 ! SIDE EFFECTS:
;  853     1381  1 !       NONE
;  854     1382  1 !
;  855     1383  1 ! USER ERRORS:
;  856     1384  1 !       NONE
;  857     1385  1 !
;  858     1386  1 !--
;  859     1387  1
;  860     1388  2 BEGIN
;  861     1389  2
;  862     1390  2 LOCAL
;  863     1391  2         CHARACTERISTIC  : VECTOR [4,WORD],       ! characteristics to set
;  864     1392  2         STATUS;
;  865     1393  2
;  866     1394  2 BIND
;  867     1395  2         !  Set up offsets into the characteristics buffer
;  868     1396  2
;  869     1397  2         BUFFER_SIZE     = CHARACTERISTIC[1] : WORD,
;  870     1398  2         DENSITY         = CHARACTERISTIC[2] : BBLOCK;
;  871     1399  2
;  872     1400  2
;  873     1401  2 ! must be at beginning of tape to set characteristics
;  874     1402  2
;  875   P 1403  2 STATUS = DO_IO (CHAN = .CHANNEL,
;  876   P 1404  2                 IOSB = IO_STATUS,
;  877     1405  2                 FUNC = IO$_REWIND);
;  878     1406  2 IF .STATUS THEN STATUS = .IO_STATUS[0];
;  879     1407  2 IF NOT .STATUS THEN ERR_EXIT (.STATUS);
;  880     1408  2
;  881     1409  2 ! read the characteristics of the tape drive
```

```
  2
P 2 STATUS = DO_IO (CHAN = .CHANNEL,
P 2                 IOSB = CHARACTERISTIC,
  2                 FUNC = IO$_SENSEMODE);
  2 IF .STATUS THEN STATUS = .CHARACTERISTIC[0];
  2 IF NOT .STATUS THEN ERR_EXIT (.STATUS);
  2
  2 ! Set up the buffer to hold the new characteristics.  Get the device
  2 !   independent stuff from the 2nd long word of IO_STATUS, use the default
  2 !   buffersize and zero the notused field
  2
  2 CHARACTERISTIC [ 0 ] = 0;
  2 BUFFER_SIZE          = .BLOCKSZ;
  2
  2 ! Now reset density to what the user specified.
  2
  2 IF .MOUNT_OPTIONS [ OPT_DENSITY ]
  2 THEN
  3     BEGIN
  3     IF .MOUNT_OPTIONS[OPT_DENS_800]
  3     THEN DENSITY[MT$V_DENSITY] = MT$K_NRZI_800
  3     ELSE
  3         IF .MOUNT_OPTIONS[OPT_DENS_1600]
  3         THEN DENSITY[MT$V_DENSITY] = MT$K_PE_1600
  3         ELSE DENSITY[MT$V_DENSITY] = MT$K_GCR_6250;
  2     END;
  2
  2 ! write the characteristics to the tape drive
  2
P 2 STATUS = DO_IO (CHAN = .CHANNEL,
P 2                 IOSB = IO_STATUS,
P 2                 FUNC = IO$_SETMODE,
  2                 P1 = CHARACTERISTIC);
  2 IF .STATUS THEN STATUS = .IO_STATUS[0];
  2 IF NOT .STATUS THEN ERR_EXIT (.STATUS);
  2
  1 END;                                          ! end of routine RESET_DENSITY


                        007C 00000 RESET_DENSITY:
                                   .WORD   Save R2,R3,R4,R5,R6              ; 1353
56      00006G   CF   9E 00002     MOVAB   CHANNEL, R6                      ;
55       0000'   CF   9E 00007     MOVAB   IO_STATUS, R5                    ;
54   00000000G   00   9E 0000C     MOVAB   LIB$STOP, R4                     ;
53   00000000G   00   9E 00013     MOVAB   COMMON_IO, R3                    ;
5E                08   C2 0001A    SUBL2   #8, SP                           ;
                  7E   7C 0001D    CLRQ    -(SP)                            ; 1405
                  7E   7C 0001F    CLRQ    -(SP)                            ;
                  7E   7C 00021    CLRQ    -(SP)                            ;
                  7E   7C 00023    CLRQ    -(SP)                            ;
                  55   DD 00025    PUSHL   R5                               ;
                  24   DD 00027    PUSHL   #36                              ;
                  66   DD 00029    PUSHL   CHANNEL                          ;
                  1A   DD 0002B    PUSHL   #26                              ;
```

```
                                63              0C   FB 0002D          CALLS   #12, COMMON_IO                                   ;
                                52              50   D0 00030          MOVL    R0, STATUS                                      ;
                                06              52   E9 00033          BLBC    STATUS, 1$                                      ; 1406
                                52              65   3C 00036          MOVZWL  IO_STATUS, STATUS                               ;
                                05              52   E8 00039          BLBS    STATUS, 2$                                      ; 1407
                                                52   DD 0003C 1$:      PUSHL   STATUS                                          ;
                                64              01   FB 0003E          CALLS   #1, LIB$STOP                                    ;
                                                7E   7C 00041 2$:      CLRQ    -(SP)                                           ; 1413
                                                7E   7C 00043          CLRQ    -(SP)                                           ;
                                                7E   7C 00045          CLRQ    -(SP)                                           ;
                                                7E   7C 00047          CLRQ    -(SP)                                           ;
                            20  AE   9F 00049          PUSHAB  CHARACTERISTIC                                                  ;
                                27   DD 0004C          PUSHL   #39                                                             ;
                                66   DD 0004E          PUSHL   CHANNEL                                                         ;
                                1A   DD 00050          PUSHL   #26                                                             ;
                                63              0C   FB 00052          CALLS   #12, COMMON_IO                                  ;
                                52              50   D0 00055          MOVL    R0, STATUS                                      ;
                                06              52   E9 00058          BLBC    STATUS, 3$                                      ; 1414
                                52              6E   3C 0005B          MOVZWL  CHARACTERISTIC, STATUS                          ;
                                05              52   E8 0005E          BLBS    STATUS, 4$                                      ; 1415
                                                52   DD 00061 3$:      PUSHL   STATUS                                          ;
                                64              01   FB 00063          CALLS   #1, LIB$STOP                                    ;
                                                6E   B4 00066 4$:      CLRW    CHARACTERISTIC                                  ; 1421
                        02      AE      F4      A5   B0 00068          MOVW    BLOCKSZ, BUFFER_SIZE                            ; 1422
                                22    0000G     CF   E9 0006D          BLBC    MOUNT_OPTIONS, 7$                               ; 1426
                    08      0000G     CF           01   E1 00072          BBC     #1, MOUNT_OPTIONS, 5$                           ; 1429
    05  AE              05              00           03   F0 00078          INSV    #3, #0, #5, DENSITY+1                           ; 1430
                                                     14   11 0007E          BRB     7$                                              ;
                    08      0000G     CF           03   E1 00080 5$:      BBC     #3, MOUNT_OPTIONS+5, 6$                         ; 1432
    05  AE              05              00           04   F0 00086          INSV    #4, #0, #5, DENSITY+1                           ; 1433
                                                     06   11 0008C          BRB     7$                                              ;
    05  AE              05              00           05   F0 0008E 6$:      INSV    #5, #0, #5, DENSITY+1                           ; 1434
                                                7E   7C 00094 7$:      CLRQ    -(SP)                                           ; 1442
                                                7E   7C 00096          CLRQ    -(SP)                                           ;
                                                7E   D4 00098          CLRL    -(SP)                                           ;
                                            14  AE   9F 0009A          PUSHAB  CHARACTERISTIC                                  ;
                                                7E   7C 0009D          CLRQ    -(SP)                                           ;
                                                55   DD 0009F          PUSHL   R5                                              ;
                                                23   DD 000A1          PUSHL   #35                                             ;
                                                66   DD 000A3          PUSHL   CHANNEL                                         ;
                                                1A   DD 000A5          PUSHL   #26                                             ;
                                63              0C   FB 000A7          CALLS   #12, COMMON_IO                                  ;
                                52              50   D0 000AA          MOVL    R0, STATUS                                      ;
                                06              52   E9 000AD          BLBC    STATUS, 8$                                      ; 1443
                                52              65   3C 000B0          MOVZWL  IO_STATUS, STATUS                               ;
                                05              52   E8 000B3          BLBS    STATUS, 9$                                      ; 1444
                                                52   DD 000B6 8$:      PUSHL   STATUS                                          ;
                                64              01   FB 000B8          CALLS   #1, LIB$STOP                                    ;
                                                     04 000BB 9$:      RET                                                     ; 1446

; Routine Size:  188 bytes,    Routine Base:  $CODE$ + 0457

;  919          1447  1                                                ! end of routine TAPE_OWN_PRO
```

```
;  921          M 1448  1 MACRO INITIALIZE_MOUNT_TAPE =
;  922          M 1449  1
;  923          M 1450  1 !++
;  924          M 1451  1 !
;  925          M 1452  1 ! FUNCTIONAL DESCRIPTION:
;  926          M 1453  1 !
;  927          M 1454  1 !       This MACRO is the code that is done 1st thru the routine MOUNT_TAPE.
;  928          M 1455  1 !       It initialize the prototypes for the MVL, RVT and VCB.  The code also
;  929          M 1456  1 !       does some one time only checks.
;  930          M 1457  1 !
;  931          M 1458  1 ! CALLING SEQUENCE:
;  932          M 1459  1 !       INITIALIZE_MOUNT_TAPE
;  933          M 1460  1 !
;  934          M 1461  1 ! PARAMETERS:
;  935          M 1462  1 !       All of MOUNT_TAPE's parameters
;  936          M 1463  1 !
;  937          M 1464  1 !--
;  938          M 1465  1
;  939          M 1466  1 BEGIN
;  940          M 1467  1
;  941          M 1468  1
;  942          M 1469  1 ! get a handle on the UCB list contained in the RVT
;  943          M 1470  1 !
;  944          M 1471  1 UCBLIST = PROTO_RVT[RVT$L_UCBLST];
;  945          M 1472  1
;  946          M 1473  1 ! Now fill in VCB prototype
;  947          M 1474  1 !
;  948          M 1475  1 PROTO_VCB[VCB$W_TRANS]    = 1;
;  949          M 1476  1 PROTO_VCB[VCB$W_MCOUNT]   = 1;
;  950          M 1477  1 PROTO_VCB[VCB$W_RECORDSZ] = .RECORDSZ;
;  951          M 1478  1
;  952          M 1479  1 ! If Files-11 use label in VOL1 else use user's label as the volume name in the
;  953          M 1480  1 ! VCB
;  954          M 1481  1 !
;  955          M 1482  1 IF .MOUNT_OPTIONS[OPT_IS_FILES11]
;  956          M 1483  1 THEN CH$COPY ( VL1$S_VOL[BL,  VOL1[VL1$T_VOLLBL],     ' ',
;  957          M 1484  1                 VCB$S_VOLNAME, PROTO_VCB[VCB$T_VOLNAME])
;  958          M 1485  1 ELSE CH$COPY ( .LABEL_STRING[0,LEN], .LABEL_STRING[0,ADDR],  ' ',
;  959          M 1486  1                 VCB$S_VOLNAME,        PROTO_VCB[VCB$T_VOLNAME]);
;  960          M 1487  1
;  961          M 1488  1 ! If Files-11 mount, fill in MVL + VCB
;  962          M 1489  1 !
;  963          M 1490  1 IF NOT ( .MOUNT_OPTIONS[OPT_FOREIGN] OR .MOUNT_OPTIONS[OPT_NOLABEL] )
;  964          M 1491  1 THEN
;  965          M 1492  1     BEGIN
;  966          M 1493  1
;  967          M 1494  1     ! stuff away the number of labels we have
;  968          M 1495  1     !
;  969          M 1496  1     IF .LABEL_COUNT EQL 0 THEN LABEL_COUNT = 1;
;  970          M 1497  1     PROTO_MVL[MVL$B_NVOLS] = .LABEL_COUNT;
;  971          M 1498  1
;  972          M 1499  1     ! copy the FILE SET ID to the MVL ( checked on tape reel switch by MTAACP )
;  973          M 1500  1     !
;  974          M 1501  1     CH$COPY ( HD1$S_FILESETID, ANSI_LABEL [ HD1$T_FILESETID ], ' ',
;  975          M 1502  1               MVL$S_SET_ID,    PROTO_MVL [ MVL$T_SET_ID ] );
;  976          M 1503  1
;  977          M 1504  1     ! copy VOL1 Accessibility Charater to MVL for default writing during
```

```
;  978          M 1505  1      !   MTAACP next volume writes
;  979          M 1506  1      !
;  980          M 1507  1      CH$MOVE (MVL$S_VOLOWNER, VOL1[VL1ST_VOLOWNER],PROTO_MVL[MVL$T_VOLOWNER]);
;  981          M 1508  1      PROTO_MVL[MVL$B_VOL_ACC] = .VOL1[VLT$B_VOLACCESS];
;  982          M 1509  1
;  983          M 1510  1      ! get a handle on the label list inside the MVL
;  984          M 1511  1      !
;  985          M 1512  1      MVL_ENTRY = PROTO_MVL+MVL$K_FIXLEN;
;  986          M 1513  1
;  987          M 1514  1      ! Fill in the known constant for the prototype VCB
;  988          M 1515  1      !
;  989          M 1516  1      PROTO_VCB[VCB$V_OVRACC]   = .MOUNT_OPTIONS[OPT_OVR_ACC];
;  990          M 1517  1      PROTO_VCB[VCB$V_OVREXP]   = .MOUNT_OPTIONS[OPT_OVR_EXP];
;  991          M 1518  1      PROTO_VCB[VCB$V_OVRLBL]   = .MOUNT_OPTIONS[OPT_OVR_ID];
;  992          M 1519  1      PROTO_VCB[VCB$V_OVRSETID] = .MOUNT_OPTIONS[OPT_OVR_SETID];
;  993          M 1520  1      PROTO_VCB[VCB$V_NOHDR3]   = .MOUNT_OPTIONS[OPT_NOHDR3];
;  994          M 1521  1      PROTO_VCB[VCB$V_OVRVOLO]  = .MOUNT_OPTIONS[OPT_OVR_VOLO];
;  995          M 1522  1      PROTO_VCB[VCB$V_INIT]     = .MOUNT_OPTIONS[OPT_INIT_ALL] OR .MOUNT_OPTIONS[OPT_INIT_CONT];
;  996          M 1523  1      PROTO_VCB[VCB$V_NOAUTO]   = .MOUNT_OPTIONS[OPT_NOAUTO];
;  997          M 1524  1      PROTO_VCB[VCB$V_INTCHG]   = .MOUNT_OPTIONS[OPT_INTERCHG];
;  998          M 1525  1
;  999          M 1526  1
; 1000          M 1527  1      PROTO_MVL[MVL$V_OPER]     = .PRIVILEGE_MASK[PRV$V_OPER];
; 1001          M 1528  1      PROTO_MVL[MVL$V_VOLPRO]   = .PRIVILEGE_MASK[PRV$V_VOLPRO];
; 1002          M 1529  1      PROTO_MVL[MVL$V_OVRPRO]   = .PRIVILEGE_MASK[PRV$V_VOLPRO] OR
; 1003          M 1530  1                                  .PRIVILEGE_MASK[PRV$V_BYPASS] OR
; 1004          M 1531  1                                  .PRIVILEGE_MASK[PRV$V_OPER]   OR
; 1005          M 1532  1                                  .PRIVILEGE_MASK[PRV$V_SYSPRV];
; 1006          M 1533  1      PROTO_MVL[MVL$B_STDVER]   = .LABEL_VER;
; 1007          M 1534  1
; 1008          M 1535  1      END;
; 1009          M 1536  1
; 1010          M 1537  1      ! must have operator privilege to monkey with the ACP
; 1011          M 1538  1      !
; 1012          M 1539  1      IF  (.MOUNT_OPTIONS[OPT_UNIQUEACP] OR
; 1013          M 1540  1           .MOUNT_OPTIONS[OPT_SAMEACP]   OR
; 1014          M 1541  1           .MOUNT_OPTIONS[OPT_FILEACP])
; 1015          M 1542  1          AND (NOT .PRIVILEGE_MASK[PRV$V_OPER])
; 1016          M 1543  1        THEN ERR_EXIT (SS$_NOPRIV);
; 1017          M 1544  1
; 1018          M 1545  1      ! If not Files-11 mount or mount foreign or mount no labels then
; 1019          M 1546  1      ! only one unit can be involved.  If Files-11 allocate
; 1020          M 1547  1      ! secondary units checking that the maximum number of devices  is not exceeded.
; 1021          M 1548  1      !
; 1022          M 1549  1      IF (.DEVICE_COUNT EQL 0)
; 1023          M 1550  1         OR ((NOT .MOUNT_OPTIONS[OPT_IS_FILES11]
; 1024          M 1551  1                OR .MOUNT_OPTIONS[OPT_FOREIGN]
; 1025          M 1552  1                OR .MOUNT_OPTIONS[OPT_NOLABEL])
; 1026          M 1553  1              AND (.DEVICE_COUNT NEQ 1))
; 1027          M 1554  1        THEN ERR_EXIT (MOUNS_DEVICES);
; 1028          M 1555  1
; 1029          M 1556  1      ! remember the first volume's UIC and Protection ( used in the UCB )
; 1030          M 1557  1      !
; 1031          M 1558  1      FIRST_V_UIC  = .VOLUME_UIC;
; 1032          M 1559  1      FIRST_V_PROT = .VOLUME_PROT;
; 1033          M 1560  1      END;
; 1034          1561  1      %;                                      ! end of Macro INITIALIZE_MOUNT_TAPE
```

```
;  1036        M 1562  1 MACRO DONE_MOUNT_TAPE =
;  1037        M 1563  1
;  1038        M 1564  1 !++
;  1039        M 1565  1 !
;  1040        M 1566  1 ! FUNCTIONAL DESCRIPTION:
;  1041        M 1567  1 !
;  1042        M 1568  1 !       This MACRO is the code that is done the last time thru the routine
;  1043        M 1569  1 !       MOUNT_TAPE.  It fills the MVL with the extra labels.  The real MVL,
;  1044        M 1570  1 !       RVT and VCB get put into system space.  The user is notified of which
;  1045        M 1571  1 !       reels are mounted where.
;  1046        M 1572  1 !
;  1047        M 1573  1 ! CALLING SEQUENCE:
;  1048        M 1574  1 !       DONE_MOUNT_TAPE
;  1049        M 1575  1 !
;  1050        M 1576  1 ! PARAMETERS:
;  1051        M 1577  1 !       All of MOUNT_TAPE's parameters
;  1052        M 1578  1 !
;  1053        M 1579  1 !--
;  1054        M 1580  1
;  1055        M 1581  1 BEGIN
;  1056        M 1582  1
;  1057        M 1583  1 ! If Files-11 mount, fill in MVL with the extra labels ( if more labels then
;  1058        M 1584  1 ! devices are specified )
;  1059        M 1585  1 !
;  1060        M 1586  1 IF NOT (.MOUNT_OPTIONS[OPT_FOREIGN] OR .MOUNT_OPTIONS[OPT_NOLABEL])
;  1061        M 1587  1 THEN
;  1062        M 1588  1     INCR I FROM (.DEVICE_INDEX + 1) TO .LABEL_COUNT - 1 DO
;  1063        M 1589  1         BEGIN
;  1064        M 1590  1         IF .LABEL_STRING [.I, LEN] GTRU VL1$S_VOLLBL
;  1065        M 1591  1           THEN ERR_EXIT (SS$_MTLBLLONG);
;  1066        M 1592  1         CH$TRANSLATE ( TRANSLATION_TABLE,
;  1067        M 1593  1                        .LABEL_STRING[.I,LEN], .LABEL_STRING[.I,ADDR], ' ',
;  1068        M 1594  1                        MVL$S_VOLLBL,          MVL_ENTRY[.I,MVL$T_VOLLBL]);
;  1069        M 1595  1         MVL_ENTRY[.I, MVL$B_STATUS]  = 0;
;  1070        M 1596  1         END;
;  1071        M 1597  1
;  1072        M 1598  1
;  1073        M 1599  1 ! update the number of units available to volume set
;  1074        M 1600  1 !
;  1075        M 1601  1 PROTO_RVT[RVT$B_NVOLS] = .DEVICE_INDEX + 1;
;  1076        M 1602  1
;  1077        M 1603  1 ! make the mount a real thing
;  1078        M 1604  1 !
;  1079        M 1605  1 STATUS = KERNEL_CALL (MAKE_TAPE_MOUNT);
;  1080        M 1606  1 IF NOT .STATUS THEN ERR_EXIT (.STATUS);
;  1081        M 1607  1
;  1082        M 1608  1 ! Let the user know if the volume has been changed to write lock
;  1083        M 1609  1 ! (ie He said the write ring was there but it wasn't )
;  1084        M 1610  1 !
;  1085        M 1611  1 IF .WRITE_RING [ 0 ] NEQ .MOUNT_OPTIONS [ OPT_WRITE ]
;  1086        M 1612  1 THEN ERR_MESSAGE ( MOUNS_WRITELOCK );
;  1087        M 1613  1
;  1088        M 1614  1
;  1089        M 1615  1 !
;  1090        M 1616  1 ! Print information message stating which volumes are mounted on which units
;  1091        M 1617  1 !
;  1092        M 1618  1
```

```
;  1093          M 1619  1 IF NOT ( .MOUNT_OPTIONS[OPT_FOREIGN] OR .MOUNT_OPTIONS[OPT_NOLABEL] )
;  1094          M 1620  1 THEN
;  1095          M 1621  1     BEGIN
;  1096          M 1622  1     LOCAL LADDR : REF VECTOR[,BYTE],
;  1097          M 1623  1           SIZE;
;  1098          M 1624  1     MVL_ENTRY = PROTO_MVL+MVL$K_FIXLEN;
;  1099          M 1625  1     INCR I FROM 0 TO .PHYS_COUNT-1 DO
;  1100          M 1626  1         INCR J FROM 0 TO .PROTO_MVL[MVL$B_NVOLS] -1 DO
;  1101          M 1627  1             IF .MVL_ENTRY[.J,MV[$B_RVN] EQL .I AND .MVL_ENTRY[.J,MVL$V_MOUNTED]
;  1102          M 1628  1             THEN
;  1103          M 1629  1                 BEGIN
;  1104          M 1630  1                 LADDR = MVL_ENTRY[.J,MVL$T_VOLLBL];
;  1105          M 1631  1                 DECR K FROM MVL$S_VOLLBL TO 0 DO
;  1106          M 1632  1                     BEGIN
;  1107          M 1633  1                     SIZE = .K;
;  1108          M 1634  1                     IF .SIZE NEQ 0
;  1109          M 1635  1                     THEN
;  1110          M 1636  1                         IF .LADDR[.SIZE-1] NEQ ' ' THEN EXITLOOP;
;  1111          M 1637  1                     END;
;  1112          M 1638  1                 ERR_MESSAGE (MOUN$_MOUNTED,3,.SIZE,MVL_ENTRY[.J,MVL$T_VOLLBL],
;  1113          M 1639  1                              PHYS_NAME[.I,LEN]);
;  1114          M 1640  1                 END;
;  1115          M 1641  1     END
;  1116          M 1642  1 ELSE
;  1117          M 1643  1     BEGIN
;  1118          M 1644  1     LOCAL LADDR : REF VECTOR[,BYTE],
;  1119          M 1645  1           SIZE;
;  1120          M 1646  1     LADDR = PROTO_VCB[VCB$T_VOLNAME];
;  1121          M 1647  1     DECR I FROM VCB$S_VOLNAME TO 0 DO
;  1122          M 1648  1         BEGIN
;  1123          M 1649  1         SIZE = .I;
;  1124          M 1650  1         IF .SIZE NEQ 0
;  1125          M 1651  1         THEN
;  1126          M 1652  1             IF .LADDR[.SIZE-1] NEQ ' ' THEN EXITLOOP;
;  1127          M 1653  1         END;
;  1128          M 1654  1     ERR_MESSAGE (MOUN$_MOUNTED,3,.SIZE,PROTO_VCB[VCB$T_VOLNAME],
;  1129          M 1655  1                  PHYS_NAME[0,LEN]);
;  1130          M 1656  1     END;
;  1131          M 1657  1 END;                                          ! end of Macro DONE_MOUNT_TAPE
;  1132            1658  1 %;
```

```
;  1134        1659  1 GLOBAL ROUTINE MOUNT_TAPE : NOVALUE  =
;  1135        1660  1
;  1136        1661  1 !++
;  1137        1662  1 !
;  1138        1663  1 ! FUNCTIONAL DESCRIPTION:
;  1139        1664  1 !
;  1140        1665  1 !       This routine performs the mechanics of mounting magnetic tape
;  1141        1666  1 !       given as input the parsed and partially validated command line.
;  1142        1667  1 !
;  1143        1668  1 ! CALLING SEQUENCE:
;  1144        1669  1 !       MOUNT_TAPES ()
;  1145        1670  1 !
;  1146        1671  1 ! INPUT PARAMETERS:
;  1147        1672  1 !       NONE
;  1148        1673  1 !
;  1149        1674  1 ! IMPLICIT INPUTS:
;  1150        1675  1 !       mount parser data base
;  1151        1676  1 !       CHANNEL channel number for I/O
;  1152        1677  1 !       VOL1 ANSI VOL1 label if Files_11
;  1153        1678  1 !
;  1154        1679  1 ! OUTPUT PARAMETERS:
;  1155        1680  1 !       NONE
;  1156        1681  1 !
;  1157        1682  1 ! IMPLICIT OUTPUTS:
;  1158        1683  1 !       NONE
;  1159        1684  1 !
;  1160        1685  1 ! ROUTINE VALUE:
;  1161        1686  1 !       NONE
;  1162        1687  1 !
;  1163        1688  1 ! SIDE EFFECTS:
;  1164        1689  1 !       VCB,RVT,MVL created
;  1165        1690  1 !
;  1166        1691  1 ! USER ERRORS:
;  1167        1692  1 !       NONE
;  1168        1693  1 !
;  1169        1694  1 !--
;  1170        1695  1
;  1171        1696  2 BEGIN
;  1172        1697  2
;  1173        1698  2 ! Define descriptor vector displacements
;  1174        1699  2 !
;  1175        1700  2 MACRO LEN       = 0,0,16,0%;
;  1176        1701  2 MACRO ADDR      = 4,0,32,0%;
;  1177        1702  2
;  1178        1703  2 EXTERNAL
;  1179        1704  2         DEVICE_COUNT,                                 ! # of devices specified
;  1180        1705  2         DEVICE_STRING   : BBLOCKVECTOR[DEVMAX,8],     ! vector of devices string
;  1181        1706  2                                                       !   descriptors
;  1182        1707  2         LABEL_STRING    : BBLOCKVECTOR[LABMAX,8],     ! vector of label string
;  1183        1708  2                                                       !   descriptors
;  1184        1709  2         PHYS_COUNT,                                   ! number of physical
;  1185        1710  2                                                       !   devices allocated
;  1186        1711  2         PHYS_NAME       : BBLOCKVECTOR[DEVMAX,8];     ! vector of physical
;  1187        1712  2                                                       !   devices allocated
;  1188        1713  2
;  1189        1714  2 LOCAL
;  1190        1715  2         STATUS,
```

```
;  1191        1716  2          UCB;
;  1192        1717  2
;  1193        1718  2      OWN
;  1194        1719  2          MVL_ENTRY       : REF BBLOCKVECTOR[LABMAX,MVL$K_LENGTH],
;  1195        1720  2          UCBLIST         : REF VECTOR;                   ! vector of UCB in RVT
;  1196        1721  2
;  1197        1722  2      ! Enable handler to clear valid on all but current device
;  1198        1723  2
;  1199        1724  2      ENABLE ERROR_HANDLER;
;  1200        1725  2
;  1201        1726  2      ! initialize things and do some 1 time checks if first time thru
;  1202        1727  2
;  1203        1728  2      IF .DEVICE_INDEX EQL 0 THEN INITIALIZE_MOUNT_TAPE;
;  1204        1729  2
;  1205        1730  2      ! Position tape to beginning again
;  1206        1731  2
;  1207      P 1732  2      STATUS = DO_IO (CHAN = .CHANNEL,
;  1208      P 1733  2                      FUNC = IO$_REWIND,
;  1209        1734  2                      IOSB = IO_STATUS);
;  1210        1735  2      IF .STATUS THEN STATUS = .IO_STATUS[0];
;  1211        1736  2      IF (NOT .STATUS) AND (.DEVICE_INDEX LSS .LABEL_COUNT) THEN ERR_EXIT (.STATUS);
;  1212        1737  2
;  1213        1738  2      ! If the accessibility routine allows us to check the VMS protection then
;  1214        1739  2      ! check privileges.
;  1215        1740  2      ! First check to see if users has read/write access to the volume. If the
;  1216        1741  2      ! user does not have access to the volume then check to see if the user
;  1217        1742  2      ! has priv's to override the access or if the user is the owner of the volume.
;  1218        1743  2
;  1219        1744  2      IF .ACCESS
;  1220        1745  2         THEN
;  1221        1746  3         BEGIN
;  1222      P 1747  3              IF NOT KERNEL_CALL (CHECK_PROT, VOLUME_PROT, VOLUME_UIC, .PROCESS_UIC,
;  1223        1748  4                                  WRITE_RING[0])
;  1224        1749  3                 THEN
;  1225        1750  4                 BEGIN
;  1226        1751  5                      IF (.MOUNT_OPTIONS[OPT_OVR_PRO]
;  1227        1752  7                         AND ( NOT (.PRIVILEGE_MASK[PRV$V_VOLPRO]
;  1228        1753  5                              OR (.VOLUME_UIC EQL .PROCESS_UIC) ) ) )
;  1229        1754  4                      THEN ERR_EXIT (SS$_NOPRIV);
;  1230        1755  3                 END;
;  1231        1756  2        END;
;  1232        1757  2
;  1233        1758  2
;  1234        1759  2      ! get the UCB of the currect channel and stuff it away in the RVT
;  1235        1760  2
;  1236        1761  2
;  1237        1762  2      UCB = KERNEL_CALL (GET_CHANNELUCB,.CHANNEL);
;  1238        1763  2
;  1239        1764  2      ! Check that duplicate device has not been specified
;  1240        1765  2
;  1241        1766  2      INCR J FROM 0 TO .DEVICE_INDEX - 1 DO
;  1242        1767  2          IF .UCBLIST[.J] EQL .UCB THEN ERR_EXIT (MOUN$_DUPDEVSPC);
;  1243        1768  2
;  1244        1769  2      UCBLIST[.DEVICE_INDEX] = .UCB;
;  1245        1770  2
;  1246        1771  2      ! If Files-11 mount, stuff the label in the MVL and mark it mounted
;  1247        1772  2
```

```
;  1248          1773  3 IF NOT ( .MOUNT_OPTIONS[OPT_FOREIGN] OR .MOUNT_OPTIONS[OPT_NOLABEL] )
;  1249          1774  2 THEN
;  1250          1775  2     BEGIN
;  1251          1776  3     CH$TRANSLATE ( TRANSLATION_TABLE,
;  1252          1777  3                    VL1$S_VOLLBL, VOL1 [VL1$T_VOLLBL], ' ',
;  1253          1778  3                    MVL$S_VOLLBL, MVL_ENTRY [.DEVICE_INDEX, MVL$T_VOLLBL]);
;  1254          1779  3     MVL_ENTRY[.DEVICE_INDEX, MVL$B_STATUS]  = 0;
;  1255          1780  3     MVL_ENTRY[.DEVICE_INDEX, MVL$V_MOUNTED] = 1;
;  1256          1781  3     MVL_ENTRY[.DEVICE_INDEX, MVL$B_RVN]     = .DEVICE_INDEX;
;  1257          1782  2     END;
;  1258          1783  2
;  1259          1784  2 ! do some last time only stuff
;  1260          1785  2
;  1261          1786  2 IF .DEVICE_INDEX EQL (.DEVICE_COUNT - 1)
;  1262          1787  2 THEN DONE_MOUNT_TAPE;
;  1263          1788  2
;  1264          1789  1 END;                                   ! end of routine MOUNT_TAPE


                                                        .PSECT  $OWN$,NOEXE,2

                                                00291   .BLKB   3
                                                00294 MVL_ENTRY:
                                                        .BLKB   4
                                                00298 UCBLIST:.BLKB   4

                                                        .EXTRN  DEVICE_COUNT, DEVICE_STRING
                                                        .EXTRN  LABEL_STRING, PHYS_COUNT
                                                        .EXTRN  PHYS_NAME

                                                        .PSECT  $CODE$,NOWRT,2

                                    0FFC 00000          .ENTRY  MOUNT_TAPE, Save R2,R3,R4,R5,R6,R7,R8,R9,-      ; 1659
                                                                R10,R11
                                5B      0000G   CF  9E  00002       MOVAB   MOUNT_OPTIONS, R11                  ;
                                5A      0000'   CF  9E  00007       MOVAB   PROTO_VCB+44, R10                   ;
                                6D      03BD    CF  DE  0000C       MOVAL   38$, (FP)                           ; 1696
                                        0000G   CF  D5  00011       TSTL    DEVICE_INDEX                        ; 1728
                                                    03  13  00015       BEQL    1$                              ;
                                                0173    31  00017       BRW     13$                             ;
                            01F8    CA      0104    CA  9E  0001A 1$:   MOVAB   PROTO_RVT+68, UCBLIST           ;
                              E0    AA              01  B0  00021       MOVW    #1, PROTO_VCB+12                ;
                              20    AA              01  B0  00025       MOVW    #1, PROTO_VCB+76                ;
                              24    AA      0000G   CF  B0  00029       MOVW    RECORDSZ, PROTO_VCB+80          ;
                    08        04    AB              01  E1  0002F       BBC     #1, MOUNT_OPTIONS+4, 2$         ;
OC          20      0000G   CF                      06  2C  00034       MOVC5   #6, VOL1+4, #32, #12, PROTO_VCB+20  ;
                                        E8      AA          0003B       ;
                                                    0B  11  0003D       BRB     3$                              ;
OC          20      0000G   DF      0000G   CF      2C  0003F 2$:       MOVC5   LABEL_STRING, @LABEL_STRING+4, #32, #12, -  ;
                                        E8      AA          00048       PROTO_VCB+20                            ;
                    03        01    AB              03  E1  0004A 3$:   BBC     #3, MOUNT_OPTIONS+1, 5$         ;
                                                00EA    31  0004F 4$:   BRW     7$                              ;
                    F8        01    AB              04  E0  00052 5$:   BBS     #4, MOUNT_OPTIONS+1, 4$         ;
                                            0000G   CF  D5  00057       TSTL    LABEL_COUNT                     ;
                                                        05  12  0005B       BNEQ    6$                          ;
                                0000G   CF              01  D0  0005D       MOVL    #1, LABEL_COUNT             ;
```

```
                        014F   CA      0000G   CF  90 00062 6$:    MOVB    LABEL_COUNT, PROTO_MVL+11                       ;
                0150    CA      FF79    CA          06  28 00069           MOVC3   #6, ANSI_LABEL+21, PROTO_MVL+12          ;
                0158    CA      0000G   CF          0E  28 00071           MOVC3   #14, VOLT+37, PROTO_MVL+20              ;
                        0156    CA      0000G   CF  90 00079           MOVB    VOL1+10, PROTO_MVL+T8                   ;
                        01F4    CA      0168    CA  9E 00080           MOVAB   PROTO_MVL+36, MVL_ENTRY                 ;
        50      04      AB              01          06  EF 00087           EXTZV   #6, #T, MOUNT_OPTIONS+4, R0             ;
        6A      01      01              01          50  F0 0008D           INSV    R0, #1, #1, PROTO_VCB+44                ;
        50      02      AB              01          04  EF 00092           EXTZV   #4, #1, MOUNT_OPTIONS+2, R0             ;
        6A      01      00              50  F0 00098           INSV    R0, #0, #1, PROTO_VCB+44                ;
        50      02      AB              01          06  EF 0009D           EXTZV   #6, #1, MOUNT_OPTIONS+2, R0             ;
        6A      01      02              50  F0 000A3           INSV    R0, #2, #1, PROTO_VCB+44                ;
        50      02      AB              01          05  EF 000A8           EXTZV   #5, #1, MOUNT_OPTIONS+2, R0             ;
        6A      01      03              50  F0 000AE           INSV    R0, #3, #1, PROTO_VCB+44                ;
        50      05      AB              01          04  EF 000B3           EXTZV   #4, #1, MOUNT_OPTIONS+5, R0             ;
        6A      01      07              50  F0 000B9           INSV    R0, #7, #1, PROTO_VCB+44                ;
        50      07      AB              01          04  EF 000BE           EXTZV   #4, #1, MOUNT_OPTIONS+7, R0             ;
01      AA      01      05              50  F0 000C4           INSV    R0, #5, #1, PROTO_VCB+45                ;
        50      07      AB              01          02  EF 000CA           EXTZV   #2, #1, MOUNT_OPTIONS+7, R0             ;
        51      07      AB              01          03  EF 000D0           EXTZV   #3, #1, MOUNT_OPTIONS+7, R1             ;
                                        50          51  88 000D6           BISB2   R1, R0                                  ;
01      AA      01      03              50  F0 000D9           INSV    R0, #3, #1, PROTO_VCB+45                ;
        50      07      AB              01          01  EF 000DF           EXTZV   #1, #1, MOUNT_OPTIONS+7, R0             ;
01      AA      01      04              50  F0 000E5           INSV    R0, #4, #1, PROTO_VCB+45                ;
        50      07      AB              01          05  EF 000EB           EXTZV   #5, #1, MOUNT_OPTIONS+7, R0             ;
        6A      01      04              50  F0 000F1           INSV    R0, #4, #1, PROTO_VCB+44                ;
                        50          CC  AA  D0 000F6           MOVL    PRIVILEGE_MASK, R0                              ;
        51      60      01              12  EF 000FA           EXTZV   #18, #1, (R0), R1                               ;
0157    CA      01      02              51  F0 000FF           INSV    R1, #2, #1, PROTO_MVL+19                        ;
        51      60      01              15  EF 00106           EXTZV   #21, #1, (R0), R1                               ;
0157    CA      01      00              51  F0 0010B           INSV    R1, #0, #1, PROTO_MVL+19                        ;
        51      60      01              15  EF 00112           EXTZV   #21, #1, (R0), R1                               ;
        52      60      01              1D  EF 00117           EXTZV   #29, #1, (R0), R2                               ;
                        51              52  C8 0011C           BISL2   R2, R1                                          ;
        53      60      01              12  EF 0011F           EXTZV   #18, #1, (R0), R3                               ;
                        51              53  C8 00124           BISL2   R3, R1                                          ;
        52      60      01              1C  EF 00127           EXTZV   #28, #1, (R0), R2                               ;
                        52              51  88 0012C           BISB2   R1, R2                                          ;
0157    CA      01      01              52  F0 0012F           INSV    R2, #1, #1, PROTO_MVL+19                        ;
                0166    CA      C8      AA  90 00136           MOVB    LABEL_VER, PROTO_MVL+34                         ;
        0A      03      AB              02  E0 0013C 7$:        BBS     #2, MOUNT_OPTIONS+3, 8$                         ;
        05      03      AB              03  E0 00141           BBS     #3, MOUNT_OPTIONS+3, 8$                         ;
        0E      03      AB              04  E1 00146           BBC     #4, MOUNT_OPTIONS+3, 9$                         ;
        09      CC      BA              12  E0 0014B 8$:        BBS     #18, @PRIVILEGE_MASK, 9$                        ;
                                        24  DD 00150           PUSHL   #36                                             ;
            00000000G   00              01  FB 00152           CALLS   #1, LIB$STOP                                    ;
                        50          0000G   CF  D0 00159 9$:        MOVL    DEVICE_COUNT, R0                        ;
                                        14  13 0015E           BEQL    11$                                             ;
        0A      04      AB              01  E1 00160           BBC     #1, MOUNT_OPTIONS+4, 10$                        ;
        05      01      AB              03  E0 00165           BBS     #3, MOUNT_OPTIONS+1, 10$                        ;
        12      01      AB              04  E1 0016A           BBC     #4, MOUNT_OPTIONS+1, 12$                        ;
                01                      50  D1 0016F 10$:        CMPL    R0, #1                                  ;
                                        0D  13 00172           BEQL    12$                                             ;
                        00728174        8F  DD 00174 11$:       PUSHL   #7504244                                        ;
            00000000G   00              01  FB 0017A           CALLS   #1, LIB$STOP                                    ;
                B8      AA      01EC    CA  D0 00181 12$:       MOVL    VOLUME_UIC, FIRST_V_UIC                         ;
                BC      AA      01E8    CA  D0 00187           MOVL    VOLUME_PROT, FIRST_V_PROT                       ;
                                        7E  7C 0018D 13$:       CLRQ    -(SP)                                           ; 1734
```

```
                                          7E  7C 0018F         CLRQ    -(SP)                                       ;
                                          7E  7C 00191         CLRQ    -(SP)                                       ;
                                          7E  7C 00193         CLRQ    -(SP)                                       ;
                                    C0    AA  9F 00195         PUSHAB  IO_STATUS                                   ;
                                          24  DD 00198         PUSHL   #36                                         ;
                                 0000G    CF  DD 0019A         PUSHL   CHANNEL                                     ;
                                          1A  DD 0019E         PUSHL   #26                                         ;
                    00000000G  00         0C  FB 001A0         CALLS   #12, COMMON_IO                              ;
                               59         50  D0 001A7         MOVL    R0, STATUS                                  ;
                               07         59  E9 001AA         BLBC    STATUS, 14$                                 ; 1735
                               59   C0    AA  3C 001AD         MOVZWL  IO_STATUS, STATUS                           ;
                               12         59  E8 001B1         BLBS    STATUS, 15$                                 ; 1736
                        0000G  CF    0000G    CF  D1 001B4 14$: CMPL   DEVICE_INDEX, LABEL_COUNT                   ;
                                          09  18 001BB         BGEQ    15$                                         ;
                                          59  DD 001BD         PUSHL   STATUS                                      ;
                    00000000G  00         01  FB 001BF         CALLS   #1, LIB$STOP                                ;
                               3B  FF60   CA  E9 001C6 15$:    BLBC    ACCESS, 16$                                 ; 1744
                                   01F0   CA  9F 001CB         PUSHAB  WRITE_RING                                  ; 1748
                                     D0   AA  DD 001CF         PUSHL   PROCESS_UIC                                 ;
                                   01EC   CA  9F 001D2         PUSHAB  VOLUME_UIC                                  ;
                                   01E8   CA  9F 001D6         PUSHAB  VOLUME_PROT                                 ;
                                          04  DD 001DA         PUSHL   #4                                          ;
                                          5E  DD 001DC         PUSHL   SP                                          ;
                                 0000G    CF  9F 001DE         PUSHAB  CHECK_PROT                                  ;
                    00000000G  9F         07  FB 001E2         CALLS   #7, @#SYS$CMKRNL                            ;
                               1A         50  E8 001E9         BLBS    R0, 16$                                     ;
                               16    04   AB  E9 001EC         BLBC    MOUNT_OPTIONS+4, 16$                        ; 1751
                 11        CC  BA         15  E0 001F0         BBS     #21, @PRIVILEGE_MASK, 16$                   ; 1752
                           D0  AA  01EC   CA  D1 001F5         CMPL    VOLUME_UIC, PROCESS_UIC                     ; 1753
                                          09  13 001FB         BEQL    16$                                         ;
                                          24  DD 001FD         PUSHL   #36                                         ; 1754
                    00000000G  00         01  FB 001FF         CALLS   #1, LIB$STOP                                ;
                                 0000G    CF  DD 00206 16$:    PUSHL   CHANNEL                                     ; 1762
                                          01  DD 0020A         PUSHL   #1                                          ;
                                          5E  DD 0020C         PUSHL   SP                                          ;
                                 0000G    CF  9F 0020E         PUSHAB  GET_CHANNELUCB                              ;
                    00000000G  9F         04  FB 00212         CALLS   #4, @#SYS$CMKRNL                            ;
                               54         50  D0 00219         MOVL    R0, UCB                                     ;
                               53  0000G  CF  D0 0021C         MOVL    DEVICE_INDEX, R3                            ; 1766
                               52         01  CE 00221         MNEGL   #1, J                                       ; 1767
                                          15  11 00224         BRB     18$                                         ;
                               54  01F8 DA42  D1 00226 17$:    CMPL    @UCBLIST[J], UCB                            ;
                                          0D  12 0022C         BNEQ    18$                                         ;
                          007280D4        8F  DD 0022E         PUSHL   #7504084                                    ;
                    00000000G  00         01  FB 00234         CALLS   #1, LIB$STOP                                ;
                 E7            52         53  F2 0023B 18$:    AOBLSS  R3, J, 17$                                  ;
                               50  0000G  CF  D0 0023F         MOVL    DEVICE_INDEX, R0                            ; 1769
                        01F8 DA40         54  D0 00244         MOVL    UCB, @UCBLIST[R0]                           ;
     57          01   AB       01         03  EF 0024A         EXTZV   #3, #1, MOUNT_OPTIONS+1, R7                 ; 1773
     50          01   AB       01         04  EF 00250         EXTZV   #4, #1, MOUNT_OPTIONS+1, R0                 ;
                               57         50  C8 00256         BISL2   R0, R7                                      ;
                               3D         57  E8 00259         BLBS    R7, 19$                                     ;
                               56  01F4   CA  D0 0025C         MOVL    MVL_ENTRY, R6                               ; 1778
                               50  0000G  CF  D0 00261         MOVL    DEVICE_INDEX, R0                            ;
                                        6640  7F 00266         PUSHAQ  (R6)[R0]                                    ;
 0000'   CF           20    0000G  CF     06  2E 00269         MOVTC   #6, VOL1+4, #32, TRANSLATION_TABLE, #6, -   ;
                               9E         06     00272                 @(SP)+                                      ;
```

```
                                                50      0000G  CF   D0 00274          MOVL    DEVICE_INDEX, R0                                        ;1779
                                                         07 A640 7F 00279          PUSHAQ  7(R6)[R0]
                                                            9E   94 0027D          CLRB    @(SP)+
                                                50      0000G  CF   D0 0027F          MOVL    DEVICE_INDEX, R0                                        ;1780
                                                         07 A640 7F 00284          PUSHAQ  7(R6)[R0]
                                                9E          01   88 00288          BISB2   #1, @(SP)+
                                                50      0000G  CF   D0 0028B          MOVL    DEVICE_INDEX, R0                                        ;1781
                                                         06 A640 7F 00290          PUSHAQ  6(R6)[R0]
                                                9E      0000G  CF   90 00294          MOVB    DEVICE_INDEX, @(SP)+
                                    50  0000G   CF          01   C3 00299 19$:       SUBL3   #1, DEVICE_COUNT, R0                                    ;1786
                                                50      0000G  CF   D1 0029F          CMPL    DEVICE_INDEX, R0
                                                            01   13 002A4          BEQL    20$
                                                                 04 002A6          RET
                                                45              57   E8 002A7 20$:       BLBS    R7, 24$                                                 ;1787
                                                58      0000G  CF   D0 002AA          MOVL    LABEL_COUNT, R8
                                                56      0000G  CF   D0 002AF          MOVL    DEVICE_INDEX, I
                                                               35   11 002B4          BRB     23$
                                                0000GCF46           7F 002B6 21$:       PUSHAQ  LABEL_STRING[I]
                                                06              9E   B1 002BB          CMPW    @(SP)+, #6
                                                                0C   1B 002BE          BLEQU   22$
                                                7E      0304    8F   3C 002C0          MOVZWL  #772, -(SP)
                                    00000000G   00              01   FB 002C5          CALLS   #1, LIB$STOP
                                                0000GCF46           7F 002CC 22$:       PUSHAQ  LABEL_STRING+4[I]
                                                50              9E   D0 002D1          MOVL    @(SP)+, R0
                                                57      01F4 DA46   7E 002D4          MOVAQ   @MVL_ENTRY[I], R7
                                                0000GCF46           7F 002DA          PUSHAQ  LABEL_STRING[I]
    0000'  CF           20                     60              9E   2E 002DF          MOVTC   @(SP)+, (R0), #32, TRANSLATION_TABLE, #6, -             ;
                                                67                  06   002E6          (R7)
                                                          07    A7   94 002E8          CLRB    7(R7)
                                                C7              56   F2 002EB 23$:       AOBLSS  R8, I, 21$
                                    00CB    CA      0000G   CF          01   81 002EF 24$:       ADDB3   #1, DEVICE_INDEX, PROTO_RVT+11
                                                                        7E   D4 002F7          CLRL    -(SP)
                                                                        5E   DD 002F9          PUSHL   SP
                                                                0000V   CF   9F 002FB          PUSHAB  MAKE_TAPE_MOUNT
                                                    00000000G   9F          03   FB 002FF          CALLS   #3, @#SYS$CMKRNL
                                                                59          50   D0 00306          MOVL    R0, STATUS
                                                                09          59   E8 00309          BLBS    STATUS, 25$
                                                                59          DD   0030C          PUSHL   STATUS
                                                    00000000G   00          01   FB 0030E          CALLS   #1, LIB$STOP
                                    50      01  AB              01          01   EF 00315 25$:       EXTZV   #1, #1, MOUNT_OPTIONS+1, R0
                                    50      01F0    CA          01          00   ED 0031B          CMPZV   #0, #1, WRITE_RING, R0
                                                                            0D   13 00322          BEQL    26$
                                                            0072A013        8F   DD 00324          PUSHL   #7512083
                                                    00000000G   00          01   FB 0032A          CALLS   #1, LIB$SIGNAL
                                                68      01  AB              03   E0 00331 26$:       BBS     #3, MOUNT_OPTIONS+1, 34$
                                                63      01  AB              04   E0 00336          BBS     #4, MOUNT_OPTIONS+1, 34$
                                                    01F4    CA      0168    CA   9E 0033B          MOVAB   PROTO_MVL+36, MVL_ENTRY
                                                57      0000G   CF          D0   00342          MOVL    PHYS_COUNT, R7
                                                52                          01   CE 00347          MNEGL   #1, I
                                                                            4D   11 0034A          BRB     33$
                                                56      014F    CA          9A   0034C 27$:       MOVZBL  PROTO_MVL+11, R6
                                                53                          01   CE 00351          MNEGL   #1, J
                                                                            3F   11 00354          BRB     32$
                                                50      01F4 DA43           7E   00356 28$:       MOVAQ   @MVL_ENTRY[J], R0
                                    52      06  A0              08          00   ED 0035C          CMPZV   #0, #8, 6(R0), I
                                                                            31   12 00362          BNEQ    32$
                                                2D          07  A0          E9   00364          BLBC    7(R0), 32$
```

```
                54           50   D0 00368          MOVL    R0, LADDR                                          ;
                51           06   D0 00368          MOVL    #6, K                                              ;
                55           51   D0 0036E 29$:     MOVL    K, SIZE                                            ;
                             07   13 00371          BEQL    30$                                                ;
                20       FF A544  91 00373          CMPB    -1(SIZE)[LADDR], #32                               ;
                             03   12 00378          BNEQ    31$                                                ;
                F1           51   F4 0037A 30$:     SOBGEQ  K, 29$                                             ;
                      0000GCF42   7F 0037D 31$:     PUSHAQ  PHYS_NAME[I]                                       ;
                             50   DD 00382          PUSHL   R0                                                 ;
                             55   DD 00384          PUSHL   SIZE                                               ;
                             03   DD 00386          PUSHL   #3                                                 ;
                    0072A003     8F   DD 00388          PUSHL   #7512067                                       ;
            00000000G   00       05   FB 0038E          CALLS   #5, LIB$SIGNAL                                 ;
BD                          53   56   F2 00395 32$:     AOBLSS  R6, J, 28$                                     ;
AF                          52   57   F2 00399 33$:     AOBLSS  R7, I, 27$                                     ;
                                 04   00 0039D          RET                                                    ;
                50       E8  AA   9E 0039E 34$:     MOVAB   PROTO_VCB+20, LADDR                                ;
                52           0C   D0 003A2          MOVL    #12, I                                             ;
                51           52   D0 003A5 35$:     MOVL    I, SIZE                                            ;
                             07   13 003A8          BEQL    36$                                                ;
                20       FF A140  91 003AA          CMPB    -1(SIZE)[LADDR], #32                               ;
                             03   12 003AF          BNEQ    37$                                                ;
                F1           52   F4 003B1 36$:     SOBGEQ  I, 35$                                             ;
                      0000G  CF   9F 003B4 37$:     PUSHAB  PHYS_NAME                                          ;
                        E8   AA   9F 003B8          PUSHAB  PROTO_VCB+20                                       ;
                             51   DD 003BB          PUSHL   SIZE                                               ;
                             03   DD 003BD          PUSHL   #3                                                 ;
                    0072A003     8F   DD 003BF          PUSHL   #7512067                                       ;
            00000000G   00       05   FB 003C5          CALLS   #5, LIB$SIGNAL                                 ;
                                 04   00 003CC          RET                                                    ; 1789
                               0000   00 003CD 38$:     .WORD   Save nothing                                   ; 1696
                             7E   D4   00 003CF          CLRL    -(SP)                                          ;
                             5E   DD   00 003D1          PUSHL   SP                                             ;
                       7E        04   AC 7D 00 003D3     MOVQ    4(AP), -(SP)                                   ;
                0000V  CF              03 FB 00 003D7     CALLS   #3, ERROR_HANDLER                              ;
                                       04 00 003DC     RET                                                    ;

; Routine Size:  989 bytes,    Routine Base:  $CODE$ + 0513

; 1265          1790  1
```

```
;  1267          1791  1 ROUTINE MAKE_TAPE_MOUNT  =
;  1268          1792  1
;  1269          1793  1 !++
;  1270          1794  1 !
;  1271          1795  1 ! FUNCTIONAL DESCRIPTION:
;  1272          1796  1 !
;  1273          1797  1 !       This routine does the data base manipulation to get a
;  1274          1798  1 !       volume mounted
;  1275          1799  1 !
;  1276          1800  1 ! CALLING SEQUENCE:
;  1277          1801  1 !       MAKE_TAPE_MOUNT (), called in kernel mode
;  1278          1802  1 !
;  1279          1803  1 ! INPUT PARAMETERS:
;  1280          1804  1 !       NONE
;  1281          1805  1 !
;  1282          1806  1 ! IMPLICIT INPUTS:
;  1283          1807  1 !       mount parser variables
;  1284          1808  1 !       own variables in this module
;  1285          1809  1 !
;  1286          1810  1 ! OUTPUT PARAMETERS:
;  1287          1811  1 !       NONE
;  1288          1812  1 !
;  1289          1813  1 ! IMPLICIT OUTPUTS:
;  1290          1814  1 !       NONE
;  1291          1815  1 !
;  1292          1816  1 ! ROUTINE VALUE:
;  1293          1817  1 !       1 - success
;  1294          1818  1 !       other status codes
;  1295          1819  1 !
;  1296          1820  1 ! SIDE EFFECTS:
;  1297          1821  1 !       NONE
;  1298          1822  1 !
;  1299          1823  1 ! USER ERRORS:
;  1300          1824  1 !       NONE
;  1301          1825  1 !
;  1302          1826  1 !--
;  1303          1827  1
;  1304          1828  2 BEGIN
;  1305          1829  2
;  1306          1830  2 EXTERNAL
;  1307          1831  2         OWNER_UIC,                              ! owner UIC from command line
;  1308          1832  2         PROTECTION,                             ! protection from command line
;  1309          1833  2         REAL_MVL        : REF BBLOCK,           ! real MVL
;  1310          1834  2         REAL_RVT        : REF BBLOCK,           ! real RVT
;  1311          1835  2         REAL_VCB        : REF BBLOCK,           ! real VCB
;  1312          1836  2         SCH$GL_CURPCB   : REF BBLOCK ADDRESSING_MODE (ABSOLUTE),
;  1313          1837  2         USER_UIC;                               ! user UIC from command line
;  1314          1838  2
;  1315          1839  2 LOCAL
;  1316          1840  2         PRIMARY_UCB     : REF BBLOCK,           ! primary UCB
;  1317          1841  2         UCB             : REF BBLOCK,           ! secondary UCB
;  1318          1842  2         PRIMARY_ORB     : REF BBLOCK,           ! primary ORB
;  1319          1843  2         ORB             : REF BBLOCK,           ! secondary ORB
;  1320          1844  2         UCBLIST         : REF VECTOR[DEVMAX];   ! UCB list in RVT
;  1321          1845  2
;  1322          1846  2 ! Enable our condition handler.
;  1323          1847  2 !
```

```
;  1324     1848  2 ENABLE KERNEL_HANDLER;
;  1325     1849  2
;  1326     1850  2 ! get the UCB of the first channel in the volume set
;  1327     1851  2 !
;  1328     1852  2 PRIMARY_UCB = .PROTO_RVT [ RVT$L_UCBLST ];
;  1329     1853  2 PRIMARY_ORB = .PRIMARY_UCB[UCB$L_ORB];
;  1330     1854  2
;  1331     1855  2 ! Setup ownership and protection
;  1332     1856  2 !
;  1333     1857  2 IF .MOUNT_OPTIONS[OPT_OWNER_UIC]
;  1334     1858  2 THEN PRIMARY_ORB[ORB$L_OWNER] = .OWNER_UIC
;  1335     1859  2 ELSE PRIMARY_ORB[ORB$L_OWNER] = .FIRST_V_UIC;
;  1336     1860  2
;  1337     1861  2 PRIMARY_ORB[ORB$V_PROT_16] = 1;                    ! SOGW protection word
;  1338     1862  2 IF .MOUNT_OPTIONS[OPT_PROTECTION]
;  1339     1863  2 THEN PRIMARY_ORB[ORB$W_PROT] = .PROTECTION<0,16> AND %X'FF00'
;  1340     1864  2 ELSE
;  1341     1865  2     IF .MOUNT_OPTIONS[OPT_FOREIGN] OR .MOUNT_OPTIONS[OPT_NOLABEL]
;  1342     1866  2     THEN PRIMARY_ORB[ORB$W_PROT] = %X'FF00'
;  1343     1867  2     ELSE PRIMARY_ORB[ORB$W_PROT] = .FIRST_V_PROT<0,16>;
;  1344     1868  2
;  1345     1869  2 ! Create real VCB
;  1346     1870  2 !
;  1347     1871  2 REAL_VCB = ALLOCATE_MEM (VCB$K_LENGTH,0);
;  1348     1872  2 REAL_VCB[VCB$B_TYPE] = DYN$C_VCB;
;  1349     1873  2 CH$MOVE (VCB$K_LENGTH-11,PROTO_VCB+11,.REAL_VCB+11);
;  1350     1874  2
;  1351     1875  2 ! If not foreign and no labels then allocate RVT and MVL
;  1352     1876  2 !
;  1353     1877  2 IF NOT .MOUNT_OPTIONS[OPT_FOREIGN] AND NOT .MOUNT_OPTIONS[OPT_NOLABEL] THEN
;  1354     1878  3     BEGIN
;  1355     1879  3     REAL_RVT                = ALLOCATE_MEM ($BYTEOFFSET (RVT$L_UCBLST) +
;  1356     1880  3                                             (.PROTO_RVT[RVT$B_NVOLS] * 4),0);
;  1357     1881  3     REAL_RVT[RVT$B_TYPE]    = DYN$C_RVT;
;  1358     1882  3     CH$MOVE (.REAL_RVT[RVT$W_SIZE]-11,PROTO_RVT+11,.REAL_RVT+11);
;  1359     1883  3     REAL_MVL                = ALLOCATE_MEM (MVL$K_FIXLEN +
;  1360     1884  3                                         (.PROTO_MVL[MVL$B_NVOLS] * MVL$K_LENGTH),0);
;  1361     1885  3     REAL_MVL[MVL$B_TYPE]    = DYN$C_MVL;
;  1362     1886  3     CH$MOVE (.REAL_MVL[MVL$W_SIZE] - 11,PROTO_MVL + 11,.REAL_MVL + 11);
;  1363     1887  3     REAL_MVL[MVL$L_VCB]     = .REAL_VCB;
;  1364     1888  3     REAL_VCB[VCB$L_RVT]     = .REAL_RVT;
;  1365     1889  3     REAL_VCB[VCB$L_MVL]     = .REAL_MVL;
;  1366     1890  3     REAL_VCB[VCB$L_BLOCKFL] = REAL_VCB[VCB$L_BLOCKFL];
;  1367     1891  3     REAL_VCB[VCB$L_BLOCKBL] = REAL_VCB[VCB$L_BLOCKFL];
;  1368     1892  3     REAL_VCB[VCB$L_VPFL]    = REAL_VCB[VCB$L_VPFL];
;  1369     1893  3     REAL_VCB[VCB$L_VPBL]    = REAL_VCB[VCB$L_VPFL];
;  1370     1894  2     END;
;  1371     1895  2
;  1372     1896  2 ALLOC_LOGNAME (0);
;  1373     1897  2
;  1374     1898  2 ! Set the "unload at dismount" characteristic in the UCB appropriately.
;  1375     1899  2 !
;  1376     1900  2 PRIMARY_UCB[UCB$V_UNLOAD] = NOT .MOUNT_OPTIONS [OPT_NOUNLOAD];
;  1377     1901  2
;  1378     1902  2 ! Check for data check requests at mount time
;  1379     1903  2 !
;  1380     1904  2 BBLOCK[PRIMARY_UCB[UCB$L_DEVCHAR],DEV$V_RCK] = .MOUNT_OPTIONS[OPT_READCHECK];
```

```
;  1381          1905  2 BBLOCK[PRIMARY_UCB[UCB$L_DEVCHAR],DEV$V_WCK] = .MOUNT_OPTIONS[OPT_WRITECHECK];
;  1382          1906  2 IF NOT .MOUNT_OPTIONS[OPT_FOREIGN] AND NOT .MOUNT_OPTIONS[OPT_NOLABEL]
;  1383          1907  2 THEN
;  1384          1908  3     BEGIN
;  1385          1909  3     PRIMARY_UCB[UCB$L_DEVCHAR] = .PRIMARY_UCB[UCB$L_DEVCHAR] AND NOT DEV$M_REC;
;  1386          1910  3     START_ACP (.PRIMARY_UCB,.REAL_VCB,AQB$K_MTA);
;  1387          1911  3     END
;  1388          1912  2 ELSE
;  1389          1913  3     BEGIN
;  1390          1914  3     LOCK_IODB ();
;  1391          1915  3     PRIMARY_UCB[UCB$L_VCB]      = .REAL_VCB;
;  1392          1916  3     PRIMARY_UCB[UCB$L_DEVCHAR] = .PRIMARY_UCB[UCB$L_DEVCHAR] OR
;  1393          1917  3                                   (DEV$M_MNT OR DEV$M_FOR OR DEV$M_REC);
;  1394          1918  3     PRIMARY_UCB[UCB$L_DEVCHAR] = .PRIMARY_UCB[UCB$L_DEVCHAR] AND
;  1395          1919  3                                   NOT (DEV$M_DIR OR DEV$M_SDI);
;  1396          1920  3     UNLOCK_IODB ();
;  1397          1921  3     END;
;  1398          1922  2
;  1399          1923  2 IF .CLEANUP_ALLOC[0] THEN PRIMARY_UCB[UCB$V_DEADMO] = 1;
;  1400          1924  2
;  1401          1925  2 IF NOT .WRITE_RING [ 0 ] THEN BBLOCK[PRIMARY_UCB[UCB$L_DEVCHAR],DEV$V_SWL] = 1;
;  1402          1926  2
;  1403          1927  2 PRIMARY_UCB [UCB$W_REFC] = .PRIMARY_UCB [UCB$W_REFC] + 1;
;  1404          1928  2
;  1405          1929  2 ! Make allocation permanent
;  1406          1930  2 !
;  1407          1931  2 !PRIMARY_UCB[UCB$B_AMOD] = 0;
;  1408          1932  2 SEND_ERRLOG (1,.PRIMARY_UCB);
;  1409          1933  2
;  1410          1934  2 ! Now set secondary UCB values if needed
;  1411          1935  2 !
;  1412          1936  2 IF .REAL_RVT NEQ 0
;  1413          1937  2 THEN
;  1414          1938  3   BEGIN
;  1415          1939  3   UCBLIST = REAL_RVT[RVT$L_UCBLST];
;  1416          1940  3   INCR I FROM 1 TO .REAL_RVT[RVT$B_NVOLS] -1 DO
;  1417          1941  4     BEGIN
;  1418          1942  4     UCB                     = .UCBLIST[.I];
;  1419          1943  4     ORB                     = .UCB[UCB$L_ORB];
;  1420          1944  4     UCB[UCB$V_UNLOAD]       = .PRIMARY_UCB[UCB$V_UNLOAD];
;  1421          1945  4 !   UCB[UCB$B_AMOD]         = 0;                ! make allocation permanent
;  1422          1946  4     ORB[ORB$L_OWNER]        = .PRIMARY_ORB[ORB$L_OWNER];
;  1423          1947  4     ORB[ORB$V_PROT_16]      = 1;
;  1424          1948  4     ORB[ORB$W_PROT]         = .PRIMARY_ORB[ORB$W_PROT];
;  1425          1949  4     UCB[UCB$L_VCB]          = .REAL_VCB;
;  1426          1950  4     UCB[UCB$W_DEVBUFSIZ]    = .PRIMARY_UCB[UCB$W_DEVBUFSIZ];
;  1427          1951  4     (UCB[UCB$[_DEVDEPEND])<0,16>
;  1428          1952  4                             = .(PRIMARY_UCB[UCB$L_DEVDEPEND])<0,16>;
;  1429          1953  4     UCB[UCB$L_DEVCHAR]      = .UCB[UCB$[_DEVCHAR] OR
;  1430          1954  4                               (DEV$M_MNT OR DEV$M_DIR OR DEV$M_SDI);
;  1431          1955  4     BBLOCK[UCB[UCB$L_DEVCHAR],DEV$V_RCK]
;  1432          1956  4                             = .BBLOCK[PRIMARY_UCB[UCB$L_DEVCHAR],DEV$V_RCK];
;  1433          1957  4     BBLOCK[UCB[UCB$L_DEVCHAR],DEV$V_WCK]
;  1434          1958  4                             = .BBLOCK[PRIMARY_UCB[UCB$L_DEVCHAR],DEV$V_WCK];
;  1435          1959  4     BBLOCK[UCB[UCB$L_DEVCHAR],DEV$V_REC]
;  1436          1960  4                             = 0;
;  1437          1961  4
```

```
;  1438          1962  4     IF .CLEANUP_ALLOC[.I] THEN UCB[UCB$V_DEADMO] = 1;
;  1439          1963  4     IF NOT .WRITE_RING [ 0 ] THEN BBLOCK[UCB[UCB$L_DEVCHAR],DEV$V_SWL] = 1;
;  1440          1964  4
;  1441          1965  4     UCB [UCB$W_REFC] = .UCB [UCB$W_REFC] + 1;
;  1442          1966  4
;  1443          1967  4     SEND_ERRLOG (1,.UCB);
;  1444          1968  3     END;
;  1445          1969  2   END;
;  1446          1970  2
;  1447          1971  2 ENTER_LOGNAME (.PRIMARY_UCB,.REAL_VCB);
;  1448          1972  2 CTL$GL_VOLUMES = .CTL$GL_VOLUMES + 1;
;  1449          1973  2
;  1450          1974  2 RETURN 1;
;  1451          1975  1 END;                                   ! end of routine MAKE_TAPE_MOUNT


                                                        .EXTRN  OWNER_UIC, PROTECTION
                                                        .EXTRN  REAL_MVL, REAL_RVT
                                                        .EXTRN  REAL_VCB, SCH$GL_CURPCB
                                                        .EXTRN  USER_UIC

                                        OFFC 00000 MAKE_TAPE_MOUNT:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11            ; 1791
                                    5B      0000G   CF  9E 00002        MOVAB   REAL_VCB, R11                   ;
                                    5A      0000G   CF  9E 00007        MOVAB   MOUNT_OPTIONS, R10              ;
                                    6D      0205    CF  DE 0000C        MOVAL   18$, (FP)                       ; 1828
                                    57      0000'   CF  D0 00011        MOVL    PROTO_RVT+68, PRIMARY_UCB       ; 1852
                                    58        1C    A7  D0 00016        MOVL    28(PRIMARY_UCB), PRIMARY_ORB    ; 1853
                            07        02    AA      02  E1 0001A        BBC     #2, MOUNT_OPTIONS+2, 1$         ; 1857
                                    68      0000G   CF  D0 0001F        MOVL    OWNER_UIC, (PRIMARY_ORB)        ; 1858
                                            05      11 00024            BRB     2$                              ;
                                    68      0000'   CF  D0 00026 1$:    MOVL    FIRST_V_UIC, (PRIMARY_ORB)      ; 1859
                                      0B    A8          01  88 0002B 2$:    BISB2   #1, 1T(PRIMARY_ORB)         ; 1861
                                0B    02    AA          01  E1 0002F        BBC     #1, MOUNT_OPTIONS+2, 3$         ; 1862
                            18  A8      0000G   CF      8F  AB 00034        BICW3   #-65281, PROTECTION, 24(PRIMARY_ORB) ; 1863
                                            18      11 0003D            BRB     6$                              ;
                                05    01    AA          03  E0 0003F 3$:    BBS     #3, MOUNT_OPTIONS+1, 4$         ; 1865
                                08    01    AA          04  E1 00044        BBC     #4, MOUNT_OPTIONS+1, 5$         ;
                                      18    A8      FF00    8F  B0 00049 4$:    MOVW    #-256, 24(PRIMARY_ORB)      ; 1866
                                            06      11 0004F            BRB     6$                              ;
                                      18    A8      0000'   CF  B0 00051 5$:    MOVW    FIRST_V_PROT, 24(PRIMARY_ORB) ; 1867
                                            7E      D4 00057 6$:        CLRL    -(SP)                           ; 1871
                                          7E    EC      8F  9A 00059        MOVZBL  #236, -(SP)                     ;
                                    0000G   CF          02  FB 0005D        CALLS   #2, ALLOCATE_MEM                ;
                                            6B          50  D0 00062        MOVL    R0, REAL_VCB                    ;
                                      0A    A0          11  90 00065        MOVB    #17, 10(R0)                     ; 1872
                            0B    A0      0000'   CF      8F  28 00069        MOVC3   #225, PROTO_VCB+11, 11(R0)      ; 1873
                                03    01    AA          03  E1 00072        BBC     #3, MOUNT_OPTIONS+1, 7$         ; 1877
                                            0083    31 00077            BRW     8$                              ;
                                7E    01    AA          04  E0 0007A 7$:    BBS     #4, MOUNT_OPTIONS+1, 8$         ;
                                            7E      D4 0007F            CLRL    -(SP)                           ; 1879
                                        50      0000'   CF  9A 00081        MOVZBL  PROTO_RVT+11, R0                ; 1880
                                7E          50          02  78 00086        ASHL    #2, R0, -(SP)               ; 1879
                                          6E  00000044      8F  C0 0008A        ADDL2   #68, (SP)                   ;
                                    0000G   CF          02  FB 00091        CALLS   #2, ALLOCATE_MEM                ;
                                    0000G   CF          50  D0 00096        MOVL    R0, REAL_RVT                    ;
```

```
                                    0A  A0              0E  90 0009B          MOVB    #14, 10(R0)                                     ; 1881
                                    51      08      A0  3C 0009F          MOVZWL  8(R0), R1                                       ; 1882
                                    51          0B  C2  000A3          SUBL2   #11, R1
                 0B  A0     0000'  CF              51  28 000A6          MOVC3   R1, PROTO_RVT+11, 11(R0)
                                                   7E  D4 000AD          CLRL    -(SP)                                           ; 1883
                                    50  0000'  CF  9A  000AF          MOVZBL  PROTO_MVL+11, R0                                ; 1884
                     7E             50          03  78 000B4          ASHL    #3, R0, -(SP)                                   ; 1885
                                    6E          24  C0 000B8          ADDL2   #36, (SP)
                            0000G   CF          02  FB 000BB          CALLS   #2, ALLOCATE_MEM
                            0000G   CF          50  D0 000C0          MOVL    R0, REAL_MVL
                                    56  0000G   CF  D0 000C5          MOVL    REAL_MVL, R6                                    ; 1885
                                    0A  A6          16  90 000CA          MOVB    #22, 10(R6)
                                    50      08  A6  3C 000CE          MOVZWL  8(R6), R0                                       ; 1886
                                    50          0B  C2  000D2          SUBL2   #11, R0
                 0B  A6     0000'  CF              50  28 000D5          MOVC3   R0, PROTO_MVL+11, 11(R6)
                                    50          6B  D0 000DC          MOVL    REAL_VCB, R0                                    ; 1887
                                    66          50  D0 000DF          MOVL    R0, (R6)
                                    20  A0  0000G   CF  D0 000E2          MOVL    REAL_RVT, 32(R0)                                ; 1888
                                    34  A0          56  D0 000E8          MOVL    R6, 52(R0)                                      ; 1889
                                    60          50  D0 000EC          MOVL    R0, (R0)                                        ; 1890
                                    04  A0          50  D0 000EF          MOVL    R0, 4(R0)                                       ; 1891
                                    3C  A0      3C  A0  9E 000F3          MOVAB   60(R0), 60(R0)                                  ; 1892
                                    40  A0      3C  A0  9E 000F8          MOVAB   60(R0), 64(R0)                                  ; 1893
                                                   7E  D4 000FD  8$:      CLRL    -(SP)                                           ; 1896
                            0000G   CF              01  FB 000FF          CALLS   #1, ALLOC_LOGNAME
         50      01  AA                 01  EF 00104          EXTZV   #2, #1, MOUNT_OPTIONS+1, R0                             ; 1900
                                        50  D2 0010A          MCOML   R0, R0
   65    A7          01                 04  F0 0010D          INSV    R0, #4, #1, 101(PRIMARY_UCB)
                                    54      38  A7  9E 00113          MOVAB   56(PRIMARY_UCB), R4                             ; 1904
         50      04  AA                 01  EF 00117          EXTZV   #3, #1, MOUNT_OPTIONS+4, R0
         64          01                 1E  F0 0011D          INSV    R0, #30, #1, (R4)
         50      04  AA                 01  EF 00122          EXTZV   #4, #1, MOUNT_OPTIONS+4, R0                             ; 1905
         64          01                 1F  F0 00128          INSV    R0, #31, #1, (R4)
                     15      01  AA             03  E0 0012D          BBS     #3, MOUNT_OPTIONS+1, 9$                         ; 1906
                     10      01  AA             04  E0 00132          BBS     #4, MOUNT_OPTIONS+1, 9$
                                    64          01  8A 00137          BICB2   #1, (R4)                                        ; 1909
                                                03  DD 0013A          PUSHL   #3                                              ; 1910
                                                6B  DD 0013C          PUSHL   REAL_VCB
                                                57  DD 0013E          PUSHL   PRIMARY_UCB
                            0000G   CF          03  FB 00140          CALLS   #3, START_ACP
                                                1C  11 00145          BRB     10$                                             ; 1906
                    00000000G   00              00  FB 00147  9$:      CALLS   #0, LOCK_IODB                                   ; 1914
                            34  A7              6B  D0 0014E          MOVL    REAL_VCB, 52(PRIMARY_UCB)                       ; 1915
                                    64  01080001  8F  C8 00152          BISL2   #17301505, (R4)                                 ; 1917
                                    64          18  8A 00159          BICB2   #24, (R4)                                       ; 1919
                    00000000G   00              00  FB 0015C          CALLS   #0, UNLOCK_IODB                                 ; 1920
                                    04      0000G   CF  E9 00163  10$:      BLBC    CLEANUP_ALLOC, 11$                      ; 1923
                            65  A7              04  88 00168          BISB2   #4, 101(PRIMARY_UCB)
                                    04      0000'   CF  E8 0016C  11$:      BLBS    WRITE_RING, 12$                         ; 1925
                            03  A4              02  88 00171          BISB2   #2, 3(R4)
                                            5C  A7  B6 00175  12$:      INCW    92(PRIMARY_UCB)                                 ; 1927
                                                57  DD 00178          PUSHL   PRIMARY_UCB                                     ; 1932
                                                01  DD 0017A          PUSHL   #1
                            0000G   CF              02  FB 0017C          CALLS   #2, SEND_ERRLOG
                                    50      0000G   CF  D0 00181          MOVL    REAL_RVT, R0                                    ; 1936
                                                7A  13 00186          BEQL    17$
                                    52      44  A0  9E 00188          MOVAB   68(R0), UCBLIST                                 ; 1939
```

```
                                59          0B    A0  9A 0018C          MOVZBL  11(R0), R9                                      ; 1940
                                            55    D4  00190            CLRL    I                                               ; 1952
                                            6A    11  00192            BRB     16$                                             ;
                                53        6245    D0  00194 13$:       MOVL    (UCBLIST)[I], UCB                               ; 1942
                                56          1C    A3  D0  00198        MOVL    28(UCB), ORB                                    ; 1943
        50          65    A7                01  04  EF  0019C        EXTZV   #4, #1, 101(PRIMARY_UCB), R0                    ; 1944
    65  A3              01                    04  50  F0  001A2        INSV    R0, #4, #1, 101(UCB)                            ;
                                66            68  D0  001A8          MOVL    (PRIMARY_ORB), (ORB)                            ; 1946
                          0B    A6            01  88  001AB          BISB2   #1, 11(ORB)                                     ; 1947
                          18    A6    18      A8  B0  001AF          MOVW    24(PRIMARY_ORB), 24(ORB)                        ; 1948
                          34    A3            6B  D0  001B4          MOVL    REAL_VCB, 52(UCB)                               ; 1949
                          42    A3    42      A7  D0  001B8          MOVL    66(PRIMARY_UCB), 66(UCB)                        ; 1950
                                50    38      A3  9E  001BD          MOVAB   56(UCB), R0                                     ; 1953
                                60  00080018  8F  C8  001C1          BISL2   #524312, (R0)                                   ; 1954
        51              64                    01  1E  EF  001C8        EXTZV   #30, #1, (R4), R1                               ; 1956
        60              01                    1E  51  F0  001CD        INSV    R1, #30, #1, (R0)                               ;
        51              64                    01  1F  EF  001D2        EXTZV   #31, #1, (R4), R1                               ; 1958
        60              01                    1F  51  F0  001D7        INSV    R1, #31, #1, (R0)                               ;
                                60            01  8A  001DC          BICB2   #1, (R0)                                        ; 1960
                        04    0000G CF        55  E1  001DF          BBC     I, CLEANUP_ALLOC, 14$                           ; 1962
                                65  A3        04  88  001E5          BISB2   #4, 101(UCB)                                    ;
                                04      0000' CF  E8  001E9 14$:     BLBS    WRITE_RING, 15$                                 ; 1963
                              03    A0        02  88  001EE          BISB2   #2, 3(R0)                                       ;
                                      5C    A3  B6  001F2 15$:       INCW    92(UCB)                                         ; 1965
                                            53  DD  001F5            PUSHL   UCB                                             ; 1967
                                            01  DD  001F7            PUSHL   #1                                              ;
                              0000G CF          02  FB  001F9        CALLS   #2, SEND_ERRLOG                                 ;
                        92                  55  F2  001FE 16$:       AOBLSS  R9, I, 13$                                      ; 1940
                                            6B  DD  00202 17$:       PUSHL   REAL_VCB                                        ; 1971
                                            57  DD  00204            PUSHL   PRIMARY_UCB                                     ;
                              0000G CF          02  FB  00206        CALLS   #2, ENTER_LOGNAME                               ;
                                  00000000G     9F  D6  0020B          INCL    @#CTL$GL_VOLUMES                                ; 1972
                                50              01  D0  00211          MOVL    #1, R0                                          ; 1974
                                                    04  00214          RET                                                     ; 1975
                                                0000    00215 18$:     .WORD   Save nothing                                    ; 1828
                                            7E  D4  00217            CLRL    -(SP)                                           ;
                                            5E  DD  00219            PUSHL   SP                                              ;
                                7E      04  AC  7D  0021B            MOVQ    4(AP), -(SP)                                    ;
                              0000V CF          03  FB  0021F        CALLS   #3, KERNEL_HANDLER                              ;
                                                    04  00224          RET                                                     ;

; Routine Size:  549 bytes,    Routine Base:  $CODE$ + 08F0
```

```
;  1453          1976  1 ROUTINE KERNEL_HANDLER (SIGNAL, MECHANISM) : NOVALUE =
;  1454          1977  1
;  1455          1978  1 !++
;  1456          1979  1 !
;  1457          1980  1 ! FUNCTIONAL DESCRIPTION:
;  1458          1981  1 !
;  1459          1982  1 !       This routine is the condition handler for all of the kernel mode
;  1460          1983  1 !       code. It undoes any damage done so far and returns the error
;  1461          1984  1 !       status to the user mode caller.
;  1462          1985  1 !
;  1463          1986  1 ! CALLING SEQUENCE:
;  1464          1987  1 !       KERNEL_HANDLER (ARG1, ARG2)
;  1465          1988  1 !
;  1466          1989  1 ! INPUT PARAMETERS:
;  1467          1990  1 !       ARG1: address of signal vector
;  1468          1991  1 !       ARG2: address of mechanism vector
;  1469          1992  1 !
;  1470          1993  1 ! IMPLICIT INPUTS:
;  1471          1994  1 !       global pointers to blocks allocated
;  1472          1995  1 !
;  1473          1996  1 ! OUTPUT PARAMETERS:
;  1474          1997  1 !       NONE
;  1475          1998  1 !
;  1476          1999  1 ! IMPLICIT OUTPUTS:
;  1477          2000  1 !       NONE
;  1478          2001  1 !
;  1479          2002  1 ! ROUTINE VALUE:
;  1480          2003  1 !       NONE
;  1481          2004  1 !
;  1482          2005  1 ! SIDE EFFECTS:
;  1483          2006  1 !       stack unwound, allocations undone
;  1484          2007  1 !
;  1485          2008  1 !--
;  1486          2009  1
;  1487          2010  2 BEGIN
;  1488          2011  2
;  1489          2012  2 MAP
;  1490          2013  2         SIGNAL          : REF BBLOCK,   ! signal vector
;  1491          2014  2         MECHANISM       : REF BBLOCK;   ! mechanism vector
;  1492          2015  2
;  1493          2016  2 LOCAL
;  1494          2017  2         UCB             : REF BBLOCK,
;  1495          2018  2         ORB             : REF BBLOCK,
;  1496          2019  2         P               : REF BBLOCK;   ! pointer to scan system lists
;  1497          2020  2
;  1498          2021  2 EXTERNAL
;  1499          2022  2         MAILBOX_CHANNEL,                ! channel number of ACP mailbox
;  1500          2023  2         REAL_VCB        : REF BBLOCK,   ! address of VCB allocated
;  1501          2024  2         REAL_RVT        : REF BBLOCK,   ! address of FCB allocated
;  1502          2025  2         REAL_MVL        : REF BBLOCK,   ! address of window allocated
;  1503          2026  2         REAL_AQB        : REF BBLOCK,   ! address of AQB allocated
;  1504          2027  2         MTL_ENTRY       : REF BBLOCK,   ! address of mounted volume list entry
;  1505          2028  2         IOC$GL_AQBLIST  : REF BBLOCK ADDRESSING_MODE (ABSOLUTE);
;  1506          2029  2                                         ! system AQB list
;  1507          2030  2
;  1508          2031  2 EXTERNAL ROUTINE
;  1509          2032  2         LOCK_IODB       : ADDRESSING_MODE (GENERAL), ! interlock system I/O database
```

```
;  1510     2033  2          UNLOCK_IODB     : ADDRESSING_MODE (GENERAL), ! unlock system I/O database
;  1511     2034  2          DEALLOCATE_MEM;                              ! deallocate system dynamic memory
;  1512     2035  2
;  1513     2036  2
;  1514     2037  2      ! Deallocate whatever control blocks exist to wherever they came from.
;  1515     2038  2      !
;  1516     2039  2
;  1517     2040  2      IF .SIGNAL[CHF$L_SIG_NAME] NEQ SS$_UNWIND
;  1518     2041  2      THEN
;  1519     2042  3          BEGIN
;  1520     2043  3
;  1521     2044  3          IF .SIGNAL[CHF$L_SIG_ARGS] NEQ 3
;  1522     2045  3          THEN BUG_CHECK (ONXSIGNAL, FATAL, 'Unexpected signal in MOUNT');
;  1523     2046  3
;  1524     2047  3      ! If there is a mailbox in existence, deassign its channel, thereby
;  1525     2048  3      ! deleting the mailbox.
;  1526     2049  3      !
;  1527     2050  3
;  1528     2051  3          IF .CLEANUP_FLAGS[CLF_DEASSMBX]
;  1529     2052  3          THEN
;  1530     2053  4              $DASSGN (CHAN = .MAILBOX_CHANNEL);
;  1531     2054  3
;  1532     2055  3      ! If we have created an AQB but no ACP, we must remove the AQB from the
;  1533     2056  3      ! system list.
;  1534     2057  3      !
;  1535     2058  3
;  1536     2059  3          IF .CLEANUP_FLAGS[CLF_DELAQB]
;  1537     2060  3          THEN
;  1538     2061  4              BEGIN
;  1539     2062  4              LOCK_IODB ();
;  1540     2063  4              P = .IOC$GL_AQBLIST;
;  1541     2064  4              IF .P EQL .REAL_AQB
;  1542     2065  4              THEN
;  1543     2066  4                  IOC$GL_AQBLIST = .REAL_AQB[AQB$L_LINK]
;  1544     2067  4              ELSE
;  1545     2068  5                  BEGIN
;  1546     2069  5                  UNTIL .P[AQB$L_LINK] EQL .REAL_AQB DO P = .P[AQB$L_LINK];
;  1547     2070  5                  P[AQB$L_LINK] = .REAL_AQB[AQB$L_LINK];
;  1548     2071  4                  END;
;  1549     2072  4              DEALLOCATE_MEM (.REAL_AQB, 0);
;  1550     2073  4              UNLOCK_IODB ();
;  1551     2074  3              END;
;  1552     2075  3
;  1553     2076  3          IF .REAL_VCB NEQ 0
;  1554     2077  3          THEN DEALLOCATE_MEM (.REAL_VCB, 0);
;  1555     2078  3
;  1556     2079  3          IF .REAL_RVT NEQ 0
;  1557     2080  3          THEN DEALLOCATE_MEM (.REAL_RVT, 0);
;  1558     2081  3
;  1559     2082  3          IF .REAL_MVL NEQ 0
;  1560     2083  3          THEN DEALLOCATE_MEM (.REAL_MVL, 0);
;  1561     2084  3
;  1562     2085  3          IF .MTL_ENTRY NEQ 0
;  1563     2086  3          THEN DEALLOCATE_MEM (.MTL_ENTRY, 1);
;  1564     2087  3
;  1565     2088  3      !
;  1566     2089  3      ! Cleanup protection on primary UCB
```

```
; 1567      2090  3 !
; 1568      2091  3
; 1569      2092  3     UCB                 = GET_CHANNELUCB (.CHANNEL);
; 1570      2093  3     ORB                 = .UCB[UCB$L_ORB];
; 1571      2094  3     ORB[ORB$L_OWNER]    = 0;
; 1572      2095  3     ORB[ORB$L_SYS_PROT] = 0;
; 1573      2096  3     ORB[ORB$L_OWN_PROT] = 0;
; 1574      2097  3     ORB[ORB$L_GRP_PROT] = 0;
; 1575      2098  3     ORB[ORB$L_WOR_PROT] = 0;
; 1576      2099  3     UCB[UCB$L_VCB]      = 0;
; 1577      2100  3
; 1578      2101  3 ! Return the condition code in R0.
; 1579      2102  3 !
; 1580      2103  3
; 1581      2104  3     MECHANISM[CHF$L_MCH_SAVR0] = .SIGNAL[CHF$L_SIG_NAME];
; 1582      2105  3     $UNWIND ();
; 1583      2106  3
; 1584      2107  2     END;
; 1585      2108  1 END;                                    ! end of routine KERNEL_HANDLER


                                                  .EXTRN  MAILBOX_CHANNEL
                                                  .EXTRN  REAL_AQB, MTL_ENTRY
                                                  .EXTRN  IOC$GL_AQBLIST, DEALLOCATE_MEM
                                                  .EXTRN  BUG$_UNXSIGNAL, SYS$DASSGN
                                                  .EXTRN  SYS$UNWIND

                                 000C 00000 KERNEL_HANDLER:
                                                  .WORD   Save R2,R3                          ; 1976
                  53 00000000G  9F   9E 00002     MOVAB   @#IOC$GL_AQBLIST, R3                ;
                  52     0000G  CF   9E 00009     MOVAB   DEALLOCATE_MEM, R2                  ;
                  50        04  AC   D0 0000E     MOVL    SIGNAL, R0                          ; 2040
     00000920     8F        04  A0   D1 00012     CMPL    4(R0), #2336                        ;
                  01            12      0001A     BNEQ    1$                                  ;
                                04      0001C     RET                                         ;
                  03            60   D1 0001D 1$: CMPL    (R0), #3                            ; 2044
                                04   13 00020     BEQL    2$                                  ;
                              FEFF      00022     BUGW                                        ; 2045
                             0000*      00024     .WORD   <BUG$_UNXSIGNAL!4>                  ;
  0B       0000G  CF            03   E1 00026 2$: BBC     #3, CLEANUP_FLAGS, 3$               ; 2051
                         0000G  CF   DD 0002C     PUSHL   MAILBOX_CHANNEL                     ; 2053
     00000000G    00            01   FB 00030     CALLS   #1, SYS$DASSGN                      ;
  39       0000G  CF            02   E1 00037 3$: BBC     #2, CLEANUP_FLAGS, 7$               ; 2059
     00000000G    00            00   FB 0003D     CALLS   #0, LOCK_IODB                       ; 2062
                  50            63   D0 00044     MOVL    IOC$GL_AQBLIST, P                   ; 2063
                  51     0000G  CF   D0 00047     MOVL    REAL_AQB, R1                        ; 2064
                  51            50   D1 0004C     CMPL    P, R1                               ;
                                06   12 0004F     BNEQ    4$                                  ;
                  63        10  A1   D0 00051     MOVL    16(R1), IOC$GL_AQBLIST              ; 2066
                                11   11 00055     BRB     6$                                  ;
                  51        10  A0   D1 00057 4$: CMPL    16(P), R1                           ; 2069
                                06   13 0005B     BEQL    5$                                  ;
                  50        10  A0   D0 0005D     MOVL    16(P), P                            ;
                                F4   11 00061     BRB     4$                                  ;
          10      A0        10  A1   D0 00063 5$: MOVL    16(R1), 16(P)                       ; 2070
                                7E   D4 00068 6$: CLRL    -(SP)                               ; 2072
```

```
                        51 DD 0006A          PUSHL   R1                                                      ;
                     62 02 FB 0006C          CALLS   #2, DEALLOCATE_MEM                                      ;
            00000000G 00 00 FB 0006F         CALLS   #0, UNLOCK_IODB                                         ; 2073
                     50 0000G CF D0 00076 7$: MOVL    REAL_VCB, R0                                            ; 2076
                        07 13 0007B          BEQL    8$                                                      ;
                        7E D4 0007D          CLRL    -(SP)                                                   ; 2077
                        50 DD 0007F          PUSHL   R0                                                      ;
                     62 02 FB 00081          CALLS   #2, DEALLOCATE_MEM                                      ;
                     50 0000G CF D0 00084 8$: MOVL    REAL_RVT, R0                                            ; 2079
                        07 13 00089          BEQL    9$                                                      ;
                        7E D4 0008B          CLRL    -(SP)                                                   ; 2080
                        50 DD 0008D          PUSHL   R0                                                      ;
                     62 02 FB 0008F          CALLS   #2, DEALLOCATE_MEM                                      ;
                     50 0000G CF D0 00092 9$: MOVL    REAL_MVL, R0                                            ; 2082
                        07 13 00097          BEQL    10$                                                     ;
                        7E D4 00099          CLRL    -(SP)                                                   ; 2083
                        50 DD 0009B          PUSHL   R0                                                      ;
                     62 02 FB 0009D          CALLS   #2, DEALLOCATE_MEM                                      ;
                     50 0000G CF D0 000A0 10$: MOVL   MTL_ENTRY, R0                                           ; 2085
                        07 13 000A5          BEQL    11$                                                     ;
                        01 DD 000A7          PUSHL   #1                                                      ; 2086
                        50 DD 000A9          PUSHL   R0                                                      ;
                     62 02 FB 000AB          CALLS   #2, DEALLOCATE_MEM                                      ;
                  0000G CF DD 000AE 11$:     PUSHL   CHANNEL                                                 ; 2092
            0000G CF 01 FB 000B2             CALLS   #1, GET_CHANNELUCB                                      ;
                     51 1C A0 D0 000B7       MOVL    28(UCB), ORB                                            ; 2093
                        61 D4 000BB          CLRL    (ORB)                                                   ; 2094
                     18 A1 7C 000BD          CLRQ    24(ORB)                                                 ; 2095
                     20 A1 7C 000C0          CLRQ    32(ORB)                                                 ; 2097
                     34 A0 D4 000C3          CLRL    52(UCB)                                                 ; 2099
                     50 04 AC 7D 000C6       MOVQ    SIGNAL, R0                                              ; 2104
                  0C A1 04 A0 D0 000CA       MOVL    4(R0), 12(R1)                                           ;
                        7E 7C 000CF          CLRQ    -(SP)                                                   ; 2105
            00000000G 00 02 FB 000D1         CALLS   #2, SYS$UNWIND                                          ;
                        04 000D8             RET                                                             ; 2108

; Routine Size:  217 bytes,    Routine Base:  $CODE$ + 0B15
```

```
;  1587          2109  1 ROUTINE ERROR_HANDLER (SIGNAL, MECHANISM)  =
;  1588          2110  1
;  1589          2111  1 !++
;  1590          2112  1 !
;  1591          2113  1 ! FUNCTIONAL DESCRIPTION:
;  1592          2114  1 !
;  1593          2115  1 !       This routine clears the valid bit for all but current UCB.
;  1594          2116  1 !
;  1595          2117  1 ! CALLING SEQUENCE:
;  1596          2118  1 !       ERROR_HANDLER ()
;  1597          2119  1 !
;  1598          2120  1 ! INPUT PARAMETERS:
;  1599          2121  1 !       NONE
;  1600          2122  1 !
;  1601          2123  1 ! IMPLICIT INPUTS:
;  1602          2124  1 !       PROTO_RVT - lists all UCB's
;  1603          2125  1 !
;  1604          2126  1 ! OUTPUT PARAMETERS:
;  1605          2127  1 !       NONE
;  1606          2128  1 !
;  1607          2129  1 ! IMPLICIT OUTPUTS:
;  1608          2130  1 !       NONE
;  1609          2131  1 !
;  1610          2132  1 ! ROUTINE VALUE:
;  1611          2133  1 !       NONE
;  1612          2134  1 !
;  1613          2135  1 ! SIDE EFFECTS:
;  1614          2136  1 !       NONE
;  1615          2137  1 !
;  1616          2138  1 !--
;  1617          2139  1
;  1618          2140  2 BEGIN
;  1619          2141  2
;  1620          2142  2 MAP
;  1621          2143  2         SIGNAL          : REF BBLOCK,   ! signal vector
;  1622          2144  2         MECHANISM       : REF BBLOCK;   ! mechanism vector
;  1623          2145  2
;  1624          2146  2 LOCAL
;  1625          2147  2         STATUS,
;  1626          2148  2         UCBLIST : REF VECTOR;
;  1627          2149  2
;  1628          2150  2 IF .BBLOCK[SIGNAL[CHF$L_SIG_NAME],STS$V_SEVERITY] EQL STS$K_SEVERE
;  1629          2151  2 THEN
;  1630          2152  3     BEGIN
;  1631          2153  3     UCBLIST = PROTO_RVT[RVT$L_UCBLST];
;  1632          2154  3     DECR I FROM .PROTO_RVT[RVT$B_NVOLS] - 2 TO 1 DO
;  1633        P 2155  3         STATUS = DO_IO ( CHAN = .CHANNEL,
;  1634        P 2156  3                          FUNC = IO$_AVAILABLE,
;  1635          2157  3                          IOSB = IO_STATUS [0]);
;  1636          2158  2     END;
;  1637          2159  2
;  1638          2160  2 RETURN SS$_RESIGNAL;
;  1639          2161  1 END;                                   ! end of routine ERROR_HANDLER
```

```
                                                0004 00000 ERROR_HANDLER:
                                                           .WORD   Save R2                                          ; 2109
                                50      04   AC  D0 00002       MOVL    SIGNAL, R0                                  ; 2150
        04      04   A0         03           00  ED 00006       CMPZV   #0, #3, 4(R0), #4                           ;
                                              2C  12 0000C       BNEQ    3$                                          ;
                                50    0000' CF  9E 0000E       MOVAB   PROTO_RVT+68, UCBLIST                       ; 2153
                                52    0000' CF  9A 00013       MOVZBL  PROTO_RVT+11, I                             ; 2154
                                              52  D7 00018       DECL    I                                           ;
                                              1B  11 0001A       BRB     2$                                          ;
                                              7E  7C 0001C 1$:   CLRQ    -(SP)                                       ; 2157
                                              7E  7C 0001E       CLRQ    -(SP)                                       ;
                                              7E  7C 00020       CLRQ    -(SP)                                       ;
                                              7E  7C 00022       CLRQ    -(SP)                                       ;
                                    0000'     CF  9F 00024       PUSHAB  IO_STATUS                                   ;
                                              11  DD 00028       PUSHL   #17                                         ;
                                    0000G     CF  DD 0002A       PUSHL   CHANNEL                                     ;
                                              1A  DD 0002E       PUSHL   #26                                         ;
                    00000000G   00            0C  FB 00030       CALLS   #12, COMMON_IO                              ;
                                E2            52  F5 00037 2$:   SOBGTR  I, 1$                                     ; 2155
                                50    0918    8F  3C 0003A 3$:   MOVZWL  #2328, R0                                  ; 2160
                                              04  0003F         RET                                                 ; 2161

; Routine Size:  64 bytes,    Routine Base:  $CODE$ + 0BEE

; 1640          2162  1
; 1641          2163  1 END
; 1642          2164  0 ELUDOM


                                                           .EXTRN  LIB$SIGNAL, LIB$STOP

;                               PSECT SUMMARY
;
;      Name                     Bytes                          Attributes
;
; $OWN$                          668  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
; $PLIT$                         292  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
; $CODE$                        3118  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                            Library Statistics
;                                      -------- Symbols --------      Pages       Processing
;      File                            Total    Loaded   Percent      Mapped      Time
;
; _$255$DUA28:[SYSLIB]LIB.L32;1         18619       156         0        1000        00:02.0
```

```
;                                       COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:MOUTAP/OBJ=OBJ$:MOUTAP MSRC$:MOUTAP/UPDATE=(ENH$:MOUTAP)

; Size:          3118 code + 960 data bytes
; Run Time:         00:58.2
; Elapsed Time:     01:42.1
; Lines/CPU Min:     2229
; Lexemes/CPU-Min: 22806
; Memory Used:   346 pages
; Compilation Complete
```

0245 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MOUPAR
LIS

MOUNTDSP
LIS

MOUNTMG
LIS

MOUTAP
LIS

RDHOME
LIS

MWTUR2
LIS

RWJMAN
LIS

MWTUR1
LIS

SRCVOL
LIS

STACP
LIS

MROBLK
LIS

REBUILD
LIS